These controllers include smooth pursuit visual tracking, inverse kinematic reaching, and operation space control of the arm [77]. This layer also provides TCP/IP interprocess communication among the Linux cluster's $1Gb$ LAN. We use the Yarp software package developed by Metta and Fitzpatrick [91]. We implemented a custom python-Yarp interface, allowing us to dynamically define and transmit data structures between processes at rates up to $100hz$. Additionally, two FireWire framegrabbers provide synchronized image pairs to the cluster. Finally, all image and sensory data are timestamped using the hardware clock from the CANbus PCI card. This ensures synchronization of the data up to the transmit time of the $1Gb$ LAN.

4.7.4 Behavior Layer

The behavior layer implements the robot's visual processing, learning, and task behaviors. These algorithms are run within our behavior-based architecture named *Slate*.

4.8 *Slate*: A Behavior Based Architecture

We have developed a behavior based architecture named *Slate*. What is meant by a robot architecture? According to Mataric [90],

> An architecture provides a principled way of organizing a control system. However, in addition to providing structure, it imposes constraints on the way the control problem can be solved.

Following Mataric, Arkin [4] notes the common aspects of behavior-based architectures:

- emphasis on the importance of coupling sensing and action tightly
- avoidance of representational symbolic knowledge
- decomposition into contextually meaningful units

Roboticists have developed many flavors of behavior based architectures. We refer to Arkin for a review [4]. Loosely stated, *Slate* is a lightweight architecture for organizing perception and control. It is implemented as a programming abstraction in Python that allows one to easily define many small computational threads. These threads can run at parameterized

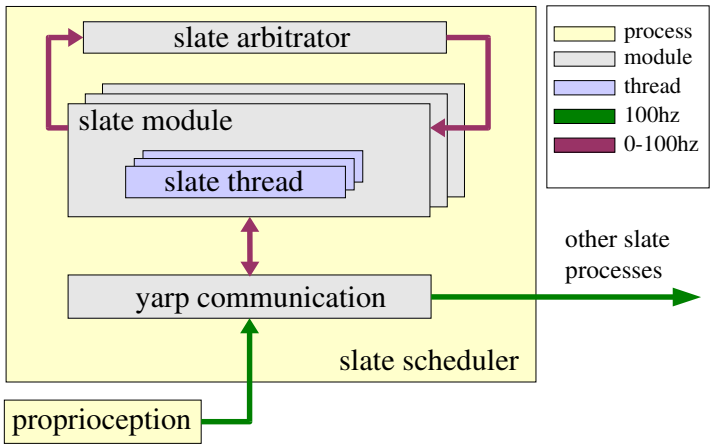


Figure 4-15: The control structure of the *Slate* architecture. Each *Slate* process runs on a node within the Linux cluster. The process starts its own non-premptive scheduler and a communication interface to external processes. At startup, the robot's proprioceptive stream is automatically imported into the process global namespace (yellow). Within a process, a module defines a namespace and a set of lightweight threads (and finite-state-automaton). These threads are scheduled at rates up to $100hz$. Threads can communicate through the global namespace or through wires. Wires provide arbitration between conflicting writes after every scheduler cycle.

rates within Slate's non-premptive scheduler. Imporantly, *Slate* makes it easy to specify time-contingent behavior and to distribute computation across multiple machines. *Slate* falls short of being a full behavior-based programming language such as Brooks's well known L-MARS[20] language but its design draws from this work. *Slate* also benefits from the use of the lightweight interpreted language, Python. Python allows for rapid development cycles and provides large toolbox of scientific, machine-learning, and vision libraries such as the open-source package pysense [73]. Computationally expensive algorithms in *Slate* can be optimized using Python extensions in C.

4.8.1 *Slate* Structure

The basic control structure of *Slate* is shown in Figure 4-15. It is built out of the following components:

Process At the highest level, *Slate* consists of many Python processes distributed across our Linux cluster. Processes pass messages back and forth at rates up to $100hz$ using TCP/IP via Yarp. Messages can be data structures of arbitrary types and can be defined on the fly. Each *Slate* process can dynamically subscribe to other *Slate* processes, image streams, motor command streams, or proprioceptive streams. By default all *Slate* processes subscribe to the robot's proprioceptive stream. A process has a global namespace where data is shared.

Scheduler A *Slate* process executes within the Linux kernel as a standard user process and it implements a non-preemptive scheduler that runs as a user thread. The scheduler is at the center of *Slate*. It's task is to schedule, and execute, short pieces of code encapsulated within the method of a Python class. Each short piece of code is associated with a *Slate* object. The scheduler maintains a list of registered *Slate* objects. Each *Slate* object has a defined update period specified in milliseconds but limited to the resolution of the scheduler's time quanta of 10 ms. Every quanta, the scheduler iterates through the list of objects, determines which are scheduled for update, and executes their update method. *Slate* objects can be threads, FSAs, and monostables. A typical process will maintain $20 - 200$ objects within the scheduler. If all scheduled objects cannot be updated within the time-window of the quanta, the actual update rate will lag the desired rate.

Module A module is an instance of a Python class. Its function is to encapsulate a group of related *Slate* objects within a shared namespace. As modules are readily parameterized, it is trivial to define multiple module instances acting on different robot limbs or perceptual streams, for example.

Thread A thread implements a small amount of computation such as processing an image frame or updating a sensor model. It is implemented as a class method of a module. A thread cannot yield and therefore the method must run within a fraction of the scheduler time quanta. The scheduler will periodically call the class method at a defined update rate. Multiple threads within a module can communicate through shared variables in the module namespace.

Port A port allows Yarp based TCP/IP interprocess communication between *Slate* mod-

ules. Connections between modules are formed through a static naming scheme. A thread can read and/or write any type of data to a port, but we assume by convention that the data read from a port is of an expected format. Ports can also connect into the robot's raw sensor, camera, and DSP controller streams.

FSA An FSA implements a time-contingent finite state automaton. Each state of the FSA is implemented as a class method of a module. The active state is periodically called by the scheduler at a defined update rate. At the end of each call, the class method optionally returns the next FSA state. Each state also has an associated timeout. If the FSA remains in a given state longer than the timeout, the active state will automatically advance to a defined next state. Optionally, an FSA will automatically reset to an idle state if it looses control of a robot resource, such as motor control of a joint.

Monostable A monostable is a simple, timing element with a defined time value, in milliseconds. A thread can trigger a monostable and it is reset to the time value. On every cycle the scheduler decrements the value by the time quanta until the value is zero. Monostables are globally accessible to all process modules.

Wire A wire is a *Slate* object that supports read/write operations. It is declared in the *Slate* global namespace and allows communication between threads and FSAs. Data written to a wire is not available for read until the next scheduler cycle, ensuring that two threads don't have inconsistent views of the same data.

Aribtrator An arbitrator resolves write conflicts on a wire. Conflicts are resolved through a mix of dynamic and fixed priorities. A fixed priority for each module is hardcoded into its definition. During each scheduler cycle, any thread can increment a module's priority by some amount. At the end of the cycle, the arbitrator grants control of the wire to the module with the highest dynamic priority. All data written to a wire by its controlling module will available to a reading thread during the next scheduler cycle. At the start of each cycle, the module's priority reverts to its fixed value, so the thread must constantly send prioritiy adjustments if it intends for a module to maintain control. Also, a hysterisis setting in the arbitrator prevents rapid behavior switching.

Tools Good development tools and libraries can make all the difference in the world when developing robots. *Slate* makes use of numerous optimized Python-C extensions for linear algebra, machine learning, and computer vision. In addition, we can easily write our own extensions using the Swig package. We have also developed useful debugging tools. A *Slate* thread can dynamically generate a remote GUI and stream data to and from it. This is an incredibly useful feature. Additionally, *Slate* can be imported into the Python runtime environment, allowing for online coding and testing of behaviors.

4.8.2 Example Program

The following Python pseudo-code demonstrates how *Slate* modules are written. It implements controller for Braitenberg vehicles 2a and 2b using the *SeekLight* and *AvoidLight* modules [14]. The modules are prioritized so the robot will drive towards a light source by default. However, if it senses too much light, then *AvoidLight* assumes control of the motor wire and the robot turns away from the light. The wire arbitrator ensures that *AvoidLigh*t is active for at least $1000ms$. The behavior threads run every $100ms$. However, the inhibit thread runs every $500ms$ because it could potentially perform heavier perceptual computation.

```
#-----------------------------------------------------
class SeekLight: #Vehicle 2b
     def motor_thread( ):
          light=slate.robot.sensors
          slate.wire_write('Wire:Motors',[light[1],light[0]])
#-----------------------------------------------------
class AvoidLight #Vehicle 2a
     def motor_thread( ):
          light=slate.robot.sensors
          slate.wire_write('Wire:Motors',[light[0],light[1]])
#-----------------------------------------------------
class MotorWriter
     def output_thread( ): #Output to controller
          desired=slate.wire_read('Wire:Motors')
```

```
            slate.robot.motors=desired
        def inhibit_thread( ): #Avoid too much light
            if (light[0]+light[1])>100:
                slate.priority_adjust('MotorArb','AvoidLight',3.0)
#-----------------------------------------------------
# Slate declarations
priority={SeekLight:2.0, AvoidLight:1.0}
a=slate.arbitrator('MotorArb', priority, hysterisis=1000)
slate.wire('Wire:Motors',arbitrator=a)
seek=slate.module(SeekLight( ))
avoid=slate.module(AvoidLight( ))
writer=slate.module(MotorWriter( ))
slate.thread(seek.motor_thread,     ms=100)
slate.thread(avoid.motor_thread,   ms=100)
slate.thread(writer.output_thread, ms=100)
slate.thread(writer.inhibit_thread, ms=500)
#-----------------------------------------------------
```

4.8.3 Describing Behaviors

Within the robotics literature, the word *behavior* has many meanings. Often, it describes a simple transformation from sensor information to motor action [4]. We avoid the difficult semantics of the word *behavior* and instead use the word *module.* The interaction of many modules with each other and the environment creates the observable behavior of Domo. A module is a well define component of *Slate.* It can be a perceptual algorithm, a motor controller, or both.

A module is always named as two or three descriptive words in italics, such as *RelaxArm* or *PersonSeek.* When appropriate, a suffix is added to signify a particular limb. For example, the *RelaxArmR* module relaxes the right arm. We find it instructive to describe the workings of a module in terms of an FSA. Within a FSA state, a module may compute an algorithm, execute a controller, or activate other modules. State transitions occur when perceptual inputs achieve a desired state or when the desired activation of other modules occurs. Following the work of Connell [33], we also emphasize that modules will often

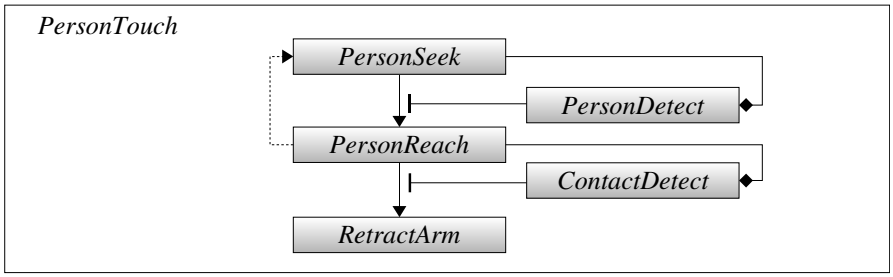


Figure 4-16: An FSA depiction of a *Slate* module. A module is described by its activation of other modules over time. While a module may employ many modules, we depict only the significant, illustrative modules as states of the FSA. Module names are italicized. A state transition (arrow) occurs contingent (bar) on perceptual feedback or the activation of another module. Often, modules will communicate *through the world* (diamond) rather than over an internal data wire. This occurs when a module takes actions that increase the likelihood of another module detecting a perceptual feature. Exceptions within a module can cause reset transitions to a previous state (dashed line). In this example, the *PersonTouch* module causes the robot to reach out and greet a person through touch. First, *PersonSeek* is activated, causing the robot to look around and increasing the likelihood of *PersonDetect*. When *PersonDetect* signals that a person is present, *PersonReach* brings the hand near the person. This action increases the likehood that *ContactDetect* will sense a person touching the arm. If *ContactDetect* is not signalled, the module reverts to search for a new person. Otherwise, *RetractArm* brings the arm down to the robot's side.

communicate through the world instead of through an internal data wire. This occurs when a module takes actions that increase the likelihood of another module detecting a perceptual feature. An example of a FSA representation is shown in Figure 4-16.

4.9 Discussion

Say something about calibration.

Architectures

Main process: 115 threads, 38 wires, 10 wire arbitrators, 32 FSAs, #yarp connections, processes total

Refer to appendix.Domo's controller is decomposed into many modular behaviors. The integration of these behaviors into a behavior-based control system allows Domo to exhibit responsive, coherent creature-like qualities. Domo's behavior based system currently includes:

Chapter 5

Visual Attention System

A visual attention system allows a robot to select for interesting, salient items from a complex world filled with numerous competeing distractions. It reduces the perceptual complexity of the environment to a small number of salient regions that can be analyzed in more detail with computationally more intensive perceptual processes. It can also provide the robot with a short-term perceptual memory through the spatial registration of visual features to a body pose-invariant ego-map.

Models of human visual attention, such as Wolfe's *Guided Search 2.0* [147], have generated design strategies for many humanoid vision systems. For example, a system developed by Breazeal [17] combines many simple image features into a single saliency map. Each feature's saliency is weighted according to the current motivation and drives of the robot. The region with the highest saliency is used to direct the gaze of the robot. In the spirit of Itti, Koch, and Niebur [67], we have implemented a visual attention system as a means to consolidate many disperate perceptual streams into a single spotlight of attention. These streams provide a continous source of simple, visual features that are relevant for cooperative manipulation tasks. Their perceptual algorithms run on different computers at different rates. Accordingly, we use a Sensory Egosphere (SES) as a consolidation point for perceptual synchronization and spatial localization.

The principal components of the visual attention system are described in Figure 5-1. In this chapter we present the visual motion model, the *InterestRegions* module, and the Sensory Egosphere.

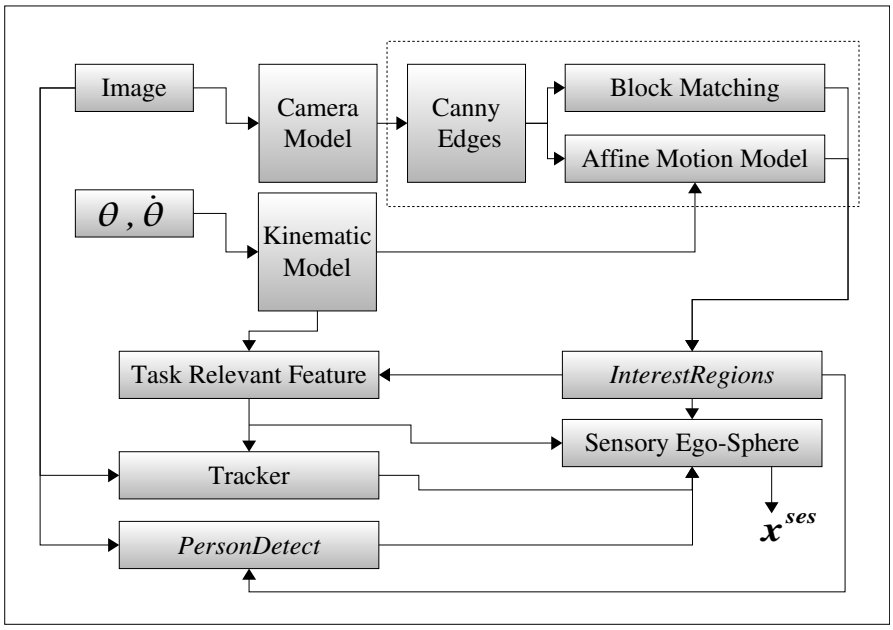


Figure 5-1: Overview of the visual attention system. The visual motion model (dashed box) selects for fast moving edges in the foreground. It can be informed by the kinematic model to discount ego-motion and to select for the robot's hand. The *InterestRegions* module detects convex shaped edges, both moving and stationary, at multiple scales. This is the principal feature that we use for detection of task relevant features for manipulation. A block-matching tracker is used to track these features during a task. The *PersonDetect* module combines a face detector, skin color model, and interest points to detect and track human features. This will be described in Chapter 7. Finally, the sensory ego-sphere consolidates the visual features into a single attention point, $\mathbf{x}^{ses}$, to direct the robot's eye gaze.

5.1 Visual Motion

Visual motion can be a robust and powerful attention cue for robot perception. For example, Fitzpatrick used motion generated by robot contact to segment an object from the background [44]. In human environments, visual motion often corresponds to objects under a person's or robot's control. However, there can be multiple sources of visual motion, such as the ego-motion of the head, a person within the environment, or motion of the manipulator. Segregation of multiple motion sources can be difficult. On Domo, we use an affine motion model to detect image points that are moving significantly with respect to the background. This model, developed by Kemp [72], is briefly reviewed in the next section. We then show how kinematic predictions of head motion and manipulator motion can be used to segregate multiple motion cues.

5.1.1 Visual Motion Model

The global image motion is estimated using a 2D affine model [125]. Image edges are then weighted based on their difference from the model's prediction. Consequently, an edge's weight reflects both the estimated speed of an edge point and its difference from the global background motion.

The global image motion, $\mathbf{A}$, is represented as a $2x3$ affine matrix that transforms a pixel location $[u_1, v_1]^T$ in image I_1 into pixel location $[u_2, v_2]^T$ in image I_2,

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix} \begin{bmatrix} u_1 \\ v_2 \\ 1 \end{bmatrix}. \tag{5.1}$$

This model can account for global changes in translation, scale, rotation, and shearing. The algorithm for estimating A is described in detail by Kemp[72]. It can be summarized as follows:

1. Find the edges in consequtive images I_1 and I_2 using a Canny edge detector [28].
2. Using the standard technique of block matching, estimate the translation $\mathbf{t}_i$ of each edge pixel between images.
3. For each translation $\mathbf{t}_i$, also compute the covariance matrix $\mathbf{C}_i$ of the block matching error .

4. Use weighted linear least squares to fit the model $\mathbf{A}$ to the translations $\mathbf{t}_i$ when weighted by $\mathbf{C}_i$.

5. Iterate the fitting process in order to remove edge points that unlikely to be part of the motion background.

6. Using the Mahalanobis distance, go back and weight each edge by how well it fits the motion model.

The algorithm generates a weighted edge map. The weight of each edge is proportional to its velocity and its difference from the global motion. Consequently, the visual motion model selects for fast moving edges in the foreground. By computing on only the edges, the model can be estimated in real-time and also reduces the use of uninformative points. The algorithm executes at approximately $20hz$ using a $1Ghz$ Pentium and $160x120$ images.

The weighted linear least squares solution can be considered the maximum likelihood estimation of the model $\mathbf{A}$ where the error is Gaussian distributed according to the covariances $\mathbf{C}_i$. Also, the Mahalanobis distance is in units of image pixels, so working with these distances is intuitive. Domo's visual attention system selects the top n edge points with the largest weights as the most salient locations in the image. If this weight is below a conservative threshold, the detection is ignored. Sample output from the algorithm is shown in Figure 5-4. Because the algorithm selects for edges moving with respect to the background, small amounts of camera motion can be ignored. This is important when working with an active vision head such as Domo's.

5.1.2 Visual Motion Prediction

Predictions can tell perceptual processes at least two things: where to look for an event and how that particular event will appear. They enable limited computational resources to perform effectively in real time by focusing the robot's attention on an expected event [40]. A robot will often know the translational and rotational velocity of its cameras and manipulators through a kinematic model. This can be used to extend our visual motion model in order to predict the perceived visual motion as the robot moves its hand or head.

Figure 5-2: Output from the visual motion model with head motion prediction (bottom) and without (top) while the camera tracks a moving person. The edge map (left) shows that the moving person is weighted more strongly when the prediction is included. The green circle shows the interest region selected by *InterestRegions* (to be described). The increased weights afforded by the prediction allow for the person's head to be selected. The optic flow estimated by the affine model is shown by the green lines (magnified $2x$).

Head Motion Prediction

We would like to find an affine matrix $\mathbf{A}_{ego}$ that describes the global optic-flow due to ego-motion of the robot's head. $\mathbf{A}_{ego}$ predicts the ego-motion of a pixel between image $I_1$and image I_2 as

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \mathbf{A}_{ego} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}.$$

The prediction can be used to inform the block-matching process in the visual motion model. The block matching estimated translation of each edge pixel by searching over an $11x11$ pixel window centered at $[u_1, v_1]^T$ in I_2. Pixel translations greater than 5 pixels cannot be matched because they fall outside of the search window. This limits the effectiveness of the model during fast camera motion. However, using $\mathbf{A_{ego}}$, we can center the search window at the predicted location $[u_2, v_2]^T$ in I_2. As shown in Figure 5-2, this allows the visual motion model to be robust during the rapid head motions that are common for an expressive, social robot head.

To estimate $\mathbf{A}_{ego}$, we first need to compute ${}^{c2}\mathbf{T}^{c1}$, the transform describing the motion of the camera between image frames I_1 and I_2. This is simply

$${}^{c2}\mathbf{T}^{c1} = {}^{c2}\mathbf{T}^{w}\,{}^{w}\mathbf{T}^{c1},$$

where ${}^{c1}\mathbf{T}^{w}$ is the world-camera transform when I_1 was captures. If head motion is not present, then ${}^{c2}\mathbf{T}^{c1} = \mathbf{I}$.

A stationary point in the world, $\mathbf{x}$, viewed from { C} over time has coordinates

$$\begin{bmatrix} x_{c1} \\ y_{c1} \\ z_{c1} \end{bmatrix} = {}^{c1}\mathbf{T}^{w}\,\mathbf{x},$$

and

$$\begin{bmatrix} x_{c2} \\ y_{c2} \\ z_{c2} \end{bmatrix} = {}^{c2}\mathbf{T}^{w}\,\mathbf{x}.$$

With a pinhole camera this corresponds to pixels $\mathbf{k}_1 = \left[\frac{fx_{c1}}{z_{c1}}, \frac{fy_{c1}}{z_{c1}}\right]$ and $\mathbf{k}_2 = \left[\frac{fx_{c2}}{z_{c2}}, \frac{fy_{c2}}{z_{c2}}\right]$. We arrive at our model for the image motion of a stationary 3D point, $\mathbf{x}$, viewed through

a perspective camera undergoing rotation and translation:

$$\begin{bmatrix} \frac{z_{c2}}{f}\mathbf{k}_2 \\ z_{c2} \\ 1 \end{bmatrix} = {}^{c2}\mathbf{T}^{c1} \begin{bmatrix} \frac{z_{c1}}{f}\mathbf{k}_1 \\ z_{c1} \\ 1 \end{bmatrix}. \tag{5.2}$$

We can make the approximation $z_{c2} \approx z_{c1}$ (weak perspective camera constraint) if the difference in depth is small compared to the average depth. For head ego-motion, this a fair assumption if we assume that the background is mostly stationary and far from the camera[72]. A characteristic of humanoid ego-motion is that the predominant source of optic flow is from camera rotation and not translation. Domo is currently stationary, so translational body motion will not create visual motion. For mobile platforms, the ego-motion estimate can be limited to periods when the body is stationary but the head is moving. Consequently,we ignore image motion resulting from camera translation. This allows for algebraic simplification of Equation 5.2 into our desired form, giving

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} \approx \begin{bmatrix} r_1 & r_2 & r_3 f \\ r_4 & r_5 & r_6 f \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}, \tag{5.3}$$

where we use the upper-left $2x3$ submatrix, $\mathbf{R}$, of ${}^{c2}\mathbf{T}^{c1}$.

Hand Motion Prediction

We now consider how to select for visual edges that correspond to the robot's moving hand. The visual motion model generates a weighted edge map, where fast moving edges in the foreground are weighted highly. However, given an affine model, $\mathbf{A}_{hand}$, of the expected hand motion in the image, we can adapt the visual motion model to select for the moving hand but ignore other objects moving in the foreground. As in the previous section, we use $\mathbf{A}_{hand}$ to inform the block-matching process. In this case however, only pixels that move according to the $\mathbf{A}_{hand}$ model will likely fall within the $11x11$ search window. Consequently, the motion of other objects in the environment, such as a person, will be naturally discounted.

There is one subtle point with this. The motion model uses the Mahalanobis distance to weight each edge by how well it fits the affine model. A large Mahalanobis distance indicates foreground motion. If $\mathbf{A}_{hand}$ is used, then a large Mahalanobis distance indicates that the edge motion does not match the expected hand motion. This is the opposite relationship

from what we desired. Consequenly we invert this relationship. If M is the matrix of Mahalanobis distances for each edge, then we use $M_{hand} = max(0, k - M)$ instead, for a constant k.

It is straightforward to estimate $\mathbf{A}_{hand}$ which has the same form as in Equation 5.1. We select a point of interest, $\mathbf{x}$, in the hand frame { H} such as a fingertip or the tip of a grasped object. When viewed from camera frame { C} over time, it has coordinates

$$\begin{bmatrix} x_{c1} \\ y_{c1} \\ z_{c1} \end{bmatrix} = {}^{c1}\mathbf{T}^{h}\,\mathbf{x},$$

and

$$\begin{bmatrix} x_{c2} \\ y_{c2} \\ z_{c2} \end{bmatrix} = {}^{c2}\mathbf{T}^{h}\,\mathbf{x}.$$

With a pinhole camera this corresponds to pixels $\mathbf{k}_1 = \left[\frac{fx_{c1}}{z_{c1}}, \frac{fy_{c1}}{z_{c1}}\right]$ and $\mathbf{k}_2 = \left[\frac{fx_{c2}}{z_{c2}}, \frac{fy_{c2}}{z_{c2}}\right]$. We are only interested in selecting edges exhibit the same translation in the image as the point. XXXHmm, center of rotation, etcXXX. This gives

$$A_{hand} = \frac{1}{f}\begin{bmatrix} f & 0 & \frac{x_{c2}}{z_{c2}} - \frac{x_{c1}}{z_{c1}} \\ 0 & f & \frac{y_{c2}}{z_{c2}} - \frac{y_{c1}}{z_{c1}} \end{bmatrix}.$$

5.2 *InterestRegions*

We saw previously that the visual motion model selects for strong moving edges in the image. These regions around these edges will often correspond to important visual features. In order to incorporate visual information distributed near strong motion edges, we use a multi-scale interest point operator developed by Kemp [72]. This algorithm is embedded in the *InterestRegions* module. We review Kemp's algorithm here.

The interest point operator detects the position and scale of significant shape features within the image. Several different shape features can be detected, including circles, parallel lines, and corners. In our work we only utilize the circular shape feature. In this case, an interest region is defined as a circular image patch at a given radius and location in the image.

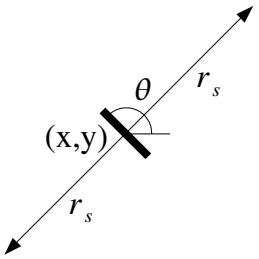


Figure 5-3: Voting by the interest point operator. This figure depicts the approximate locations in the image of the two votes at scale s cast by an edge with orientation θ and position (x, y) (Reproduced, with permission, from Kemp [72]).

Traditional interest point methods, often characterized as blob-detectors, rely on constant contrast within an interest region. However, Kemp's method is an edge-based approach, making it compatible with the weighted edge map generated by our visual motion model. Kemp's algorithm has similarities to classic image processing techniques such as the distance transform, medial axis transform, and hough transform for circles [47].

The input to the interest point detector consists of a set of weighted edges, e_i, where each edge i consists of a weight, w_i, an image location, $\mathbf{x}_i$, and an angle, θ_i. These are provided by our visual motion model. Each scale space s corresponds to a circle of a given radius r_s in pixels. For a given scale space s, each edge votes on two locations that correspond with the centers of the coarse circular regions that the edge borders. As depicted in Figure 5-3, the two votes are approximately at distance r_s from the edge's location and are located in positions orthogonal to the edge's length. It is assumed that the angle θ_i denotes the direction of the edge's length and is in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$, so that no distinction is made between the two sides of the edge.

For each scale s there is a 2D histogram that accumulates votes for interest points. The discretization of these histograms is determined by the integer bin length, l_s. The bin

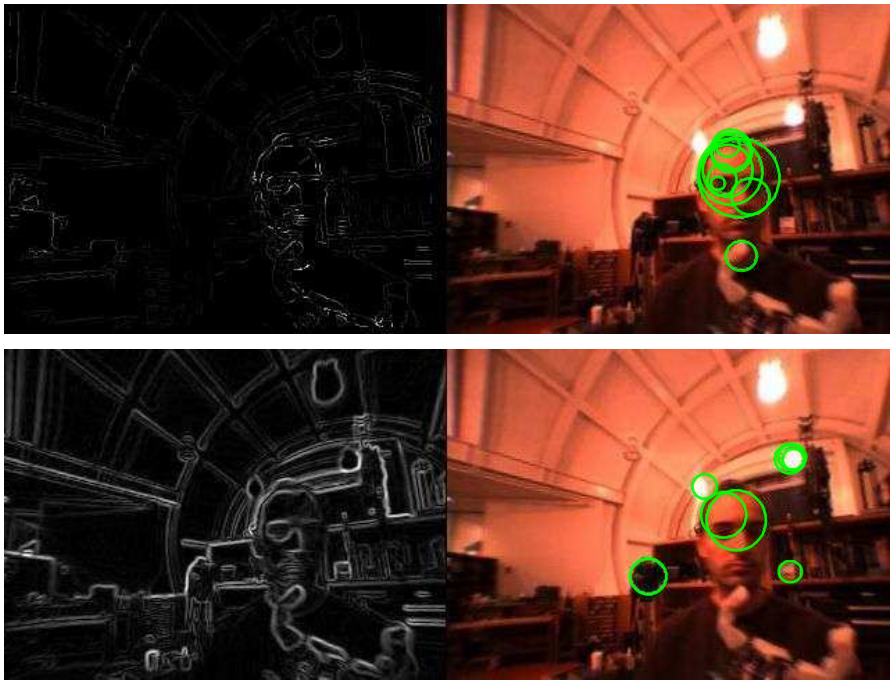


Figure 5-4: Output from the visual motion model and interest point operators. The motion weighted edge map (top, left) and unweighted edge map (bottom, left) are fed to the multi-scale interest point operator. The green circles indicate the location and size of circular interest regions. Weighting the edges by the foreground motion localizes the interest regions on salient features such as a person's head and the tip of the robot's finger (top, right).

indices, (b_x, b_y), for the histogram at scale s are computed as

$$b_s(\mathbf{x}, \theta) = \text{round}(\frac{1}{l_s}(\mathbf{x} + r_s \begin{bmatrix} \cos(\theta + \frac{\pi}{2}) \\ \sin(\theta + \frac{\pi}{2}) \end{bmatrix})), \tag{5.4}$$

which adds a vector of length r_s to the edge position $\mathbf{x}$ and then scales and quantizes the result to find the appropriate bin in the histogram. Now, the edges are iterated over and their weighted contributions added to the appropriate bins. This results in the interest point detection maps, m_s. In order to soften the effects of the block discretization, each 2D histogram, m_s, is low-pass filtered with a separable, truncated, FIR Gaussian. Finally, for each scale-space s, the point within m_s with the highest response is selects as the interest region for that particular scale. This interest region is made available to the visual attention system.

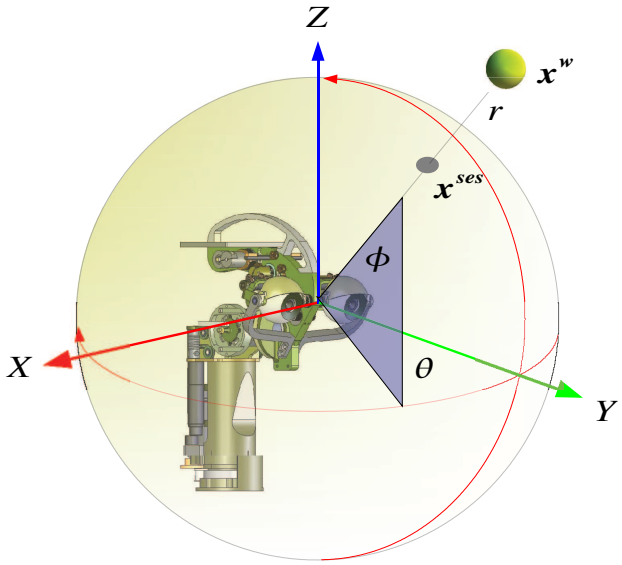


Figure 5-5: The sensory ego-sphere.

The algorithm is computationally efficient and suitable for real-time processing. When combined with the motion model, it can robustly select for significant moving shape features within the image. It can also be used without the motion model ($w_i = 1$) to detect circular edges in a static image. Sample output from the algorithm is shown in Figure 5-4.

5.3 The Sensory Ego-Sphere

The sensory ego-sphere (SES) is short-term memory mechanism for a robot. The notion originated with Albus [3] and was further developed by Peters [62]. The idea is quite simple and is illustrated in Figure 5-5. Perceptual features which may be sensed in different modalities, coordinate frames, and at different times can be brought into a single, spherical coordinate frame centered on the robot. In doing this, the data can be fused according to its spatio-temporal coincidence. The SES retains the spatial location of the data when the robot redirects its attention to other locations. For visual features, this involves transforming from pixel coordinates to ego-spherical coordinates. If a visual feature is stationary in the world while the head is moving, then the feature will be stationary in the ego-sphere frame though not in the image frame. Thus, the SES provides a stable frame of reference for perception.

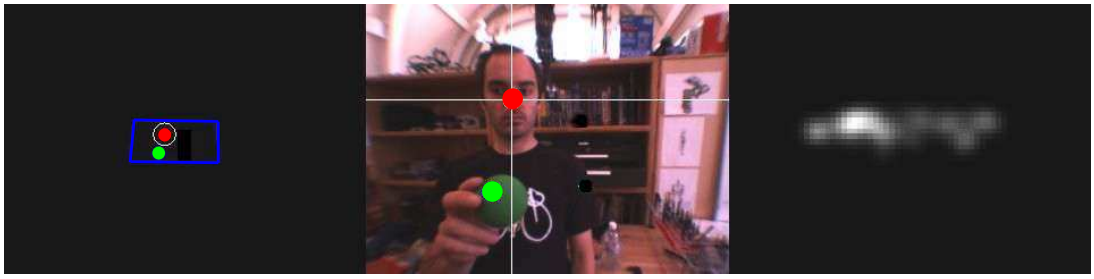


Figure 5-6: View of the sensory ego-sphere (SES). **Left:** View of the SES showing the detection of a face (red) and a fiducial (green). The camera's field-of-view projected into the SES is shown in blue. **Middle**: The detected features in the image. **Right**: The spatial distribution of face detection features over time within the SES.

This allows for stable visual servo control of the head when a tracked feature is detected intermittently or at a low rate.

Domo uses a simplified version of Peters's SES formulation. The robot torso is assumed to be stationary with respect to the world. As shown in Figure 4-1, we define the frame $\{SES\}$ as a translation of the world frame $\{W\}$ such that its origin is at the midpoint between the robot's eyes when it is looking straight ahead. It shares the same orientation as frame $\{W\}$. A point in the SES has spherical coordinates $\mathbf{x}^{ses} = [\theta, \phi, r]$, where r is the radius, θ is yaw in the transverse plane, and ϕ is pitch in the sagittal plane. The SES is defined in the $[\frac{-\pi}{2} : \frac{\pi}{2}]$ hemisphere with $[0, 0]$ pointing straight ahead. If the translation from $\{W\}$ to $\{SES\}$ is $\mathbf{t}^{ses}_{worg}$, then we project a point in the world $\mathbf{x}^{w}$ into the SES by $\mathbf{x}^{ses} = f_{sph}(\mathbf{x}^{w} - \mathbf{t}^{ses}_{worg})$, where f_{sph} maps from cartesian to spherical coordinates,

$$\begin{aligned} \mathbf{x}^{ses} = f_{sph}([x, y, z]^T) &= [\theta, \phi, r]^T \\ r &= \left(x^2 + y^2 + z^2\right)^{\frac{1}{2}} \\ \theta &= \arctan\left(\frac{x}{z}\right) \\ \phi &= \arcsin\left(\frac{y}{r}\right). \end{aligned}$$

In order to project an image pixel into the SES without pixel depth information, we assume a fixed pixel depth of $100cm$ and use a pinhole camera model to find its world coordinates. This assumes that the visual effects of head translation are small compared to rotation which is true for points far from the camera. We visualize the SES in Figure 5-6.

5.3.1 Features in the SES

The SES serves as a consolidation point for several types of features, including:

- A person's face
- A person's waving hand
- One of its hands
- The tip of a grasped object
- A randomly selected target
- The most likely location to see a face
- A colored fiducial

The perceptual algorithms for the non-trivial of these features will be described in coming chapters. Feature detections are added to the SES at rates from $10 - 30hz$. We low-pass filter a feature's location, allowing the head to smoothly track a feature even if it disappears momentarily or a false detection occurs. Each feature has a monostable timeout of $1s$, after which it is removed from the SES if a new detection has not been recieved. At any time, the output from the SES is a target in spherical coordinates, $\mathbf{x}^{ses}$, which is used to direct the robot's gaze. This target is selected using a *Slate* aribtrator and modules compete to direct the robot's gaze to a particular feature. For example, a *WatchHand* module can direct the gaze to the robot's hand if it has the highest dynamic priority of all writers to the arbitrator.

5.3.2 Spatial Distributions in the SES

We can accumulated evidence over time of a visual feature's location in the SES. Some features will appear in predictable regions. For example, Figure 5-6 visualizes the spatial distribution of face detections over many hours. As we will see in Chapter 7, such prior information can allow the visual attention system to ignore unlikely feature detections. Torralba [133] describes a framework that uses similar information to improve visual search and modulate the saliency of image regions.

We can model the spatial distribution of an image feature as the probability distribution $p(\mathbf{x}^{ses}|f)$. This represents the chance of seeing feature f at location $\mathbf{x}^{ses} = [\theta, \phi, r]$ in the SES. The distribution is estimated using a 2D histogram over $[\theta, \phi]$. Each dimension of the histogram maps to a $[-\frac{\pi}{2}, \frac{\pi}{2}]$ range of the corresponding dimension of of the SES. The distribution is updated whenever the feature appears and it is saved to disk when the robot is not running, allowing the robot to estimate the distribution over a long period of time.

We estimate $p(\mathbf{x}^{ses}|f)$ for each visual feature, f, using $100x100$ bin 2D histograms with

$$p(\mathbf{x}^{ses}|f) \approx \frac{1}{\sum_{\mathbf{d}\in D_f} w(\mathbf{d})} \sum_{\mathbf{d}\in D_f} w(\mathbf{d})\delta(\text{round}(\frac{100}{\pi}(\mathbf{d} - \mathbf{x}^{ses}))), \tag{5.5}$$

where $\delta(d) = \begin{cases} 1 \text{ if } d = 0 \\ 0 \text{ otherwise} \end{cases}$ and D_f is the accumulated detections of f in the SES.

We use a weighting function $w(\mathbf{d})$ that reduces the influence of older detections. When a new detection $\mathbf{d}_i$ is added to the distribution its weight is initialized at $w(\mathbf{d}_i) = 1.0$. This value is decremented at a rate of Δw_f per second. Detection $\mathbf{d}_i$ is removed from D_f if $w(\mathbf{d_i}) < 0.01$. This allows the distribution to adapt to recent changes such as an object changing location.

Chapter 6

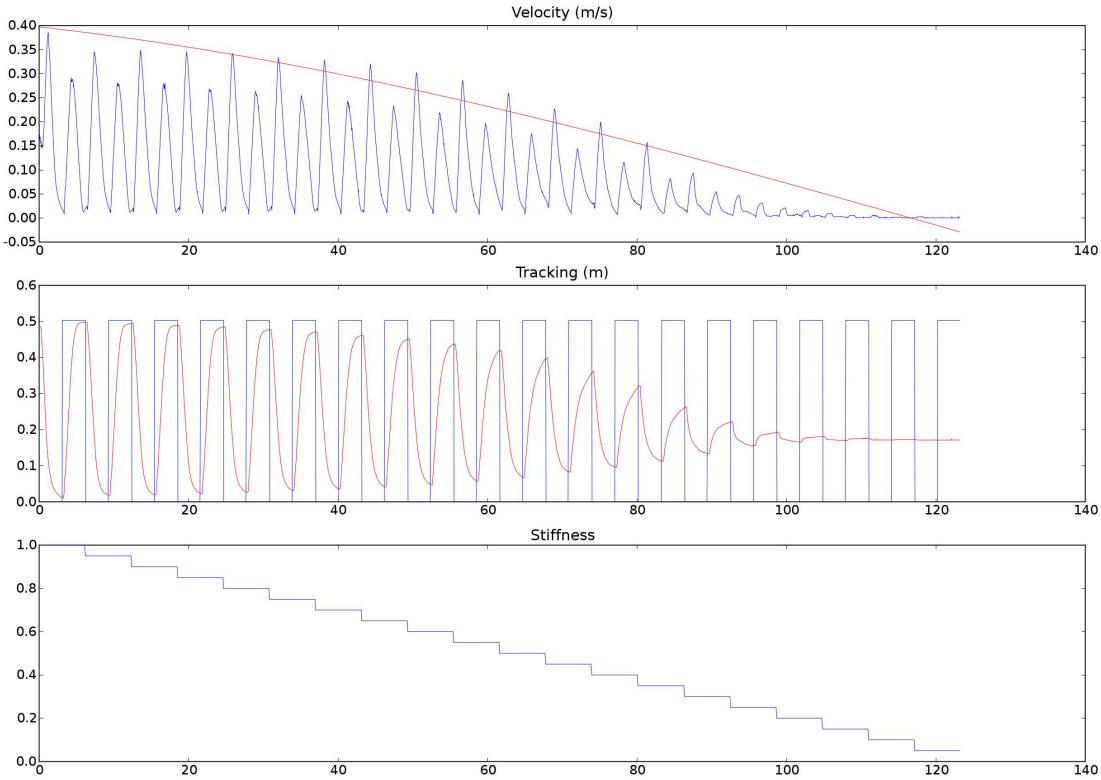
Let the Body do the Thinking

In this chapter we describe a group of *Slate* modules that leverage the robot's physical embodiment. *StiffnessAdapt* allows a module to control the manipulator impedance during a task. *ContactDetect* is triggered when contact is made with the world. The *GraspAperture* module estimates the diameter of a grasp given the proprioceptive state of the hand. These simpler modules are then combined into the *SurfaceTest* and *SurfacePlace* modules. These modules are examples of compensatory behaviors where the robot takes actions to reduce its uncertainty about the world. *SurfaceTest* allows Domo to reach out and verify the uncertain location of a hypothesized flat surface. *SurfacePlace* exploits the robot's compliance to place unknown objects upright on a surface.

6.1 *StiffnessAdapt*

Despite its simplicity, the *StiffnessAdapt* module is an effective tool for dealing with perceptual uncertainty. By lowering the stiffness of the arm, *StiffnessAdapt* trades off precision for compliance. Lowered arm stiffness is advantageous when contact with an unknown surface is anticipated. It allows the hand to maintain contact and adapt its posture to the unknown surface. Typically, actuator saturation prevents a manipulator from responding with low impedance to unexpected contact [83]. However, the springs in Domo's actuators allow the manipulator to exhibit a low effective impedance above the control bandwidth of the actuator.

A stiffness for each joint of the arms and hands is specified by the controller parameter $0 \leq K_{ps} \leq 1$ from Equation 4.4. This DSP controller essentially simulates a virtual torsion



The effect of manipulator stiffness on controller response. The hand was servoed between two targets 0.5 meters apart every 3 seconds. (Bottom) The arm stiffness was ramped from $K_{ps} = 1.0$ to $K_{ps} = 0.0$. (Middle) Blue indicates the distance of the desired hand location from the first target. Red indicates the hand's distance from the desired target. (Top) Blue indicates the magnitude of the hand's velocity. The red line illustrates the learned decision boundary for the *ContactDetect* module (to be described).

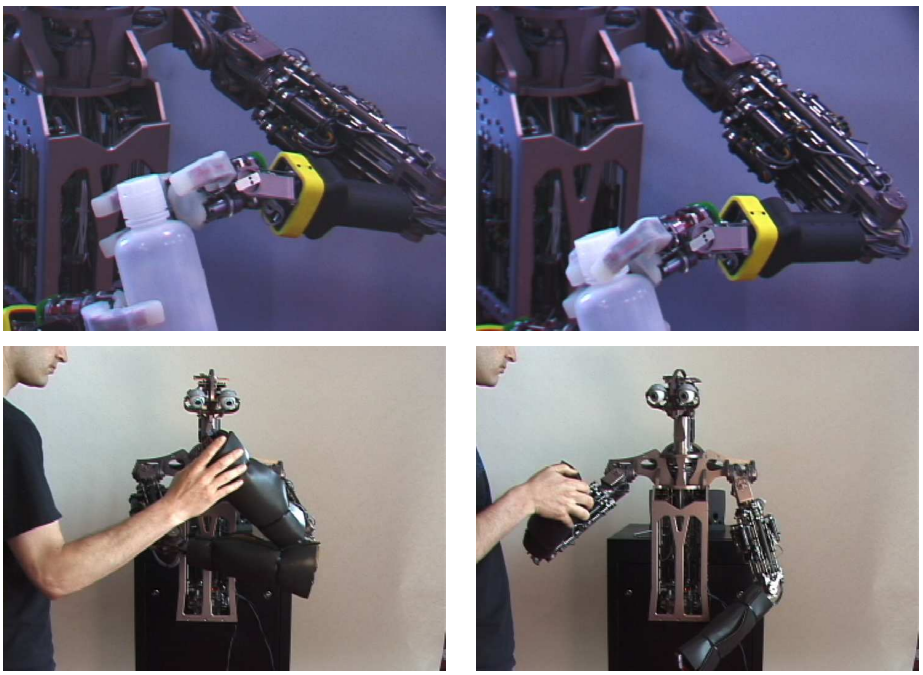


Figure 6-1: Detecting manipulator contact using low impedance and using a dynamic model. (Top) A bottle is held by the right hand and the right arm has low impedance. When contact forces generated by the left hand cause the right hand to to move, *ContactDetect* triggers a grasp reflex. (Bottom) When the arm has high impedance, torques generated by human contact violate the prediction of a dynamic model. This triggers the *WatchHand* module to direct the eye gaze to the robot's hand.

spring at the joint with stiffness K_{ps}. Figure 6.1 shows the effect of varying K_{ps} on the controller response. As stiffness is lowered, the manipulator position control performance degrades. However, an integral term in the controller, as well as secondary control loops such as visual servoing, can allow for precise but low-impedance control of the arm. The *StiffnessAdapt* simply provides arbitration among competing modules that specify the stiffness of the arm. The request with the highest priority, as described in Section 4.8, is then transmitted to the DSP controller. Ideally, a module would learn a desired arm stiffness for a task, or use sensory feedback to adapt the stiffness. In our work, the joint stiffness requested by a module is determined experimentally.

6.2 *ContactDetect*

The *ContactDetect* module detects when the manipulator makes contact with the world. As shown in Figure 6-1, two different methods are employed. In the first, low manipulator impedance is used to transform contact forces in to detectable motion at the hand. In the second, high manipulator impedance allows contact forces to generate detectable errors in a dynamic model. *ContactDetect* determines which detector to use based on the manipulator impedance commanded by *StiffnessAdapt*.

6.2.1 Contact Motion

This method of contact detection simply monitors the stiffness of the manipulator and the velocity of its hand. When the arm has low impedance, we expect that disturbance forces will cause unexpected hand motion. As shown in Figure 6.1, the expected maximum velocity of the hand, v_{max}, is a function of the manipulator stiffness. The stiffness is defined by the controller gain K_{ps} while the magnitude of the instantaneous hand velocity is $\left\|\mathbf{J}\dot{\mathbf{\Theta}}\right\|$, where the Jacobian $\mathbf{J}$ converts joint rates to a Cartesian velocity at the hand.

We used support vector regression (SVR) to learn the function $G_v(\cdot)$ such that $v_{max} \approx G_v(K_{ps})$ [30]. First, the hand was servoed between two targets 0.5 meters apart every 3 seconds for two minutes. Simultaneously, the arm stiffness was ramped from $K_{ps} = 1.0$ to $K_{ps} = 0.0$. For each value of K_{ps}, we collected $v_{max} = max(\left\|\mathbf{J}\dot{\mathbf{\Theta}}\right\|)$. The SVR was trained on each pairing of K_{ps} and v_{max}. The top row of Figure 6.1 plots $G_v(K_{ps})$.

ContactDetect signals that external contact has been made when $\left\|\mathbf{J}\dot{\mathbf{\Theta}}\right\| - G_v(K_{ps})$ is above a threshold. This method is best suited for low manipulator impedances where the disturbance velocity of the hand is large given contact. It is susceptible to false-positives when the impedance is high. Consequently, contact is ignored when $K_{ps} > 0.5$.

6.2.2 Contact Forces

The joint-space form of manipulator dynamics is

$$\tau_{dyn} = M(\mathbf{\Theta})\ddot{\mathbf{\Theta}} + V(\mathbf{\Theta}, \dot{\mathbf{\Theta}}) + G(\mathbf{\Theta}),$$

where $M(\mathbf{\Theta})\ddot{\mathbf{\Theta}}$ is the torque due to mass accelerations, $V(\mathbf{\Theta}, \dot{\mathbf{\Theta}})$ is the centrifugal and Coriolis torque, and $G(\mathbf{\Theta})$ is the torque due to gravity [35]. Given this model, the error

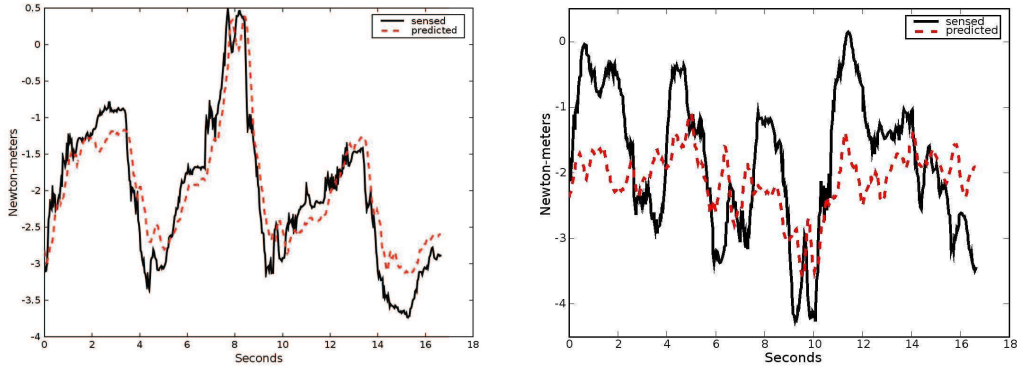


Figure 6-2: (Left) The torque prediction error for the shoulder pitch joint during non-contact reaching. (Right) The same measurement while a person makes contact with the arm during reaching.

between the predicted joint torques, τ_{dyn}, and the sensed torques, τ_{sense}, can be used to detect manipulator contact with the world.

A complete model requires estimating the mass distribution of the arms. In practice, dynamic models can be difficult to obtain and calibrate. In addition, the joint acceleration is difficult to measure precisely due to sensor resolution and anti-aliasing errors. However, the presence of external contact forces can be detected using some common model simplifications. We assume that $V(\mathbf{\Theta}, \dot{\mathbf{\Theta}}) = 0$ and approximate the inertia tensor of $M(\mathbf{\Theta})$ using a point mass for the robot forearm and bicep. Using the recursive Newton-Euler formulation, we predict the joint torques as $\tau_{dyn} = M(\mathbf{\Theta})\ddot{\mathbf{\Theta}} + G(\mathbf{\Theta})$ [52].

The model error, defined as $\tau_{dyn} - \tau_{sense}$, is used to signal contact. As shown in Figure 6-2, the error is large when manipulator contact is made. However, during dynamic reaching, errors result from our model simplifications. Therefore, we distinguish between errors induced by contact and those induced by unmodelled dynamics.

This is done by building an error histogram for each joint and each type of error. The arm executed reaching movements sampled across its workspace. At $10hz$ the error (Nm) between the error $\tau_{dyn} - \tau_{sense}$ was measured. Error histograms were accumulated for both non-contact reaching and when contact disturbances were applied by a person. Figure 6-3 shows these histograms. We see that contact errors are largest for the first two joints (pitch,yaw) of the shoulder. For the other joints, the two error types are difficult to distin-

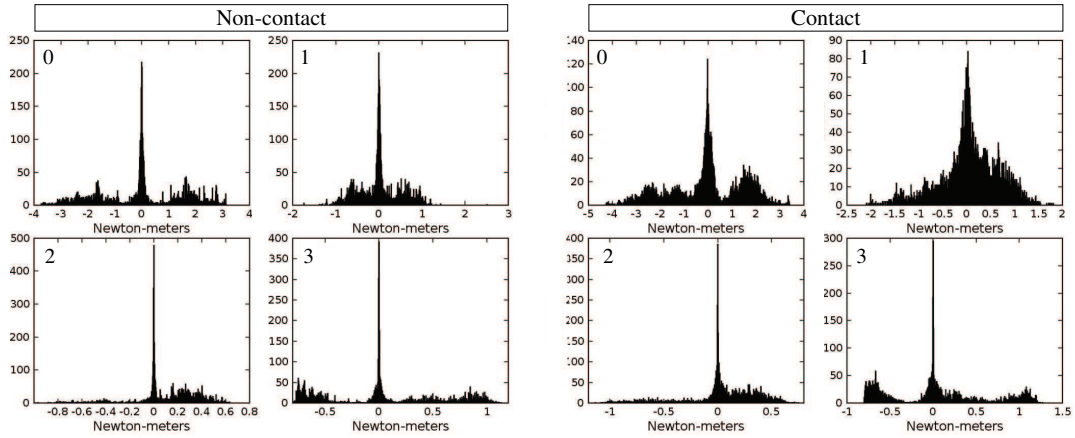


Figure 6-3: Error histograms for the torque prediction of the first four joints. The arm executed reaching movements sampled across its workspace. At $10hz$ the error (Nm) between the predicted and sensed joint torque was measured. Error distributions were measured for both non-contact reaching (Left) and when contact disturbances were applied by a person during reaching (Right). During normal operation, contact is signaled for a joint when the measured error is unlikely given the non-contact distribution. Contact errors are largest for the first two joints (pitch,yaw) of the shoulder.

guish. Consequently, only the shoulder joints are used signal contact. Contact is signaled when the magnitude of the prediction error is above a threshold and it is unlikely the error is due to dynamic reaching (as measured by the error histograms).

6.3 *GraspAperture* and *GraspDetect*

The grasp aperture, typically defined as the distance between the thumb and forefinger, is a common measure used when studying human manipulation. Prior to grasping an object, a person's grasp aperture varies according to the object being grasped and the perceptual uncertainty about the object [87]. On a robot, the grasp aperture can be used to estimate the size of an unknown, grasped object. For example, the grasp aperture created by a power grasp on a cylinder is proportional to the cylinder diameter. Ideally, the grasp aperture is measured directly using a kinematic model of the robot's hand. However this can be difficult due to non-ideal joint sensing, unmodelled kinematics, and compliant effects in the robot's finger and skin. Also, the distance between the thumb and forefinger is not always a good measure of object size as this distance can vary depending on the grasp. For some grasps, these digits may not even make contact with the object.

On Domo, there is substantial compliance in the finger. As shown in Figure 4-10, the compliant fingertip and skin allow the finger surface to deform during grasping. However, this deformation cannot be measured directly as it occurs between the joint angle sensor and the object. Consequently, it would be difficult to measure the grasp aperture kinematically without modeling the complex effects the compliance.

Instead of building a complex hand model, a map was learned between the four joint angles of the hand and the grasp aperture. Training data was gathered for 50 trial power grasps formed on five cylindrical objects of known diameters between 25 and 75mm. Because the power grasp is force controlled, we are able to manually apply forces to the object, causing it to be displaced from its equillibrium pose. For limited displacements, the fingers will maintain contact with the object. In this way, we determined the possible joint postures that result in a stable grasp for each object. As each object was displaced, the joint angles of the fingers were recorded. We used support vector regression (SVR) with a Gaussian RBF kernel to learn the function relating the cylinder diameters and the joint angles. This function, $G_a(\mathbf{\Theta})$, predicts the diameter of a grasped cylinder given the joint configuration

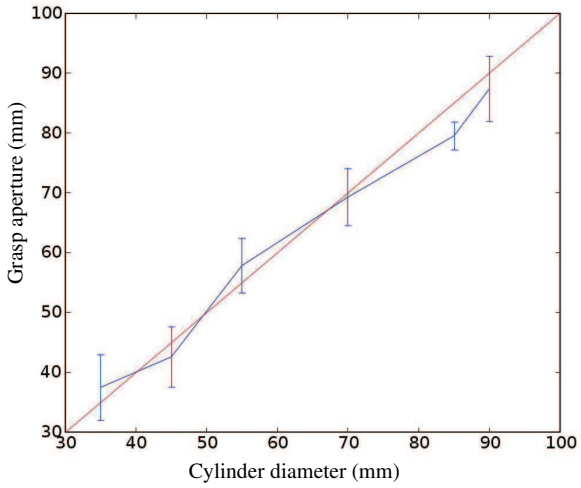


Figure 6-4: The predicted grasp aperture $G_a(\mathbf{\Theta})$ (Y axis) versus the diameter of a known, grasped cylinder (X axis). For each of six test cylinders, the object was grasped using force control. The object was then manually moved about the full grasp workspace that permitted all three fingers to remain in contact. The error bars show the maximum extent of the predicted aperture around its mean.

$\mathbf{\Theta}$.

In Figure 6-4 we show the predicted grasp aperture for a known objects not in the training set. For each of six cylinders, the object was grasped and manually moved about the set of postures that permitted all three fingers to remain in contact. We see that $G_a(\mathbf{\Theta})$ can predict the cylinder size within $10mm$ across the set of stable grasps. In fact, the natural resting pose of the hand does even better. As the object is moved far from this pose, the peformance degrades. Also, we would expect this approach to readily extend to hands with greater kinematic complexity. The *GraspAperture* module simply computes $G_a(\mathbf{\Theta})$ in real-time during task execution. As we will see, modules such as *SurfacePlace* use the estimated grasp aperture to adapt their behavior.

Related to *GraspAperature* is *GraspDetect,* which signals that a stable grasp has been made on an object. As in the work of Connell [33], *GraspDetect* is informed by the sensory state of the hand in the world rather than the internal state of the grasp controller. It relies on three conditions to detect a grasp. First, it monitors the net torque applied by the fingers. If it is positive (closing) and above a threshold, then it is assumed that the

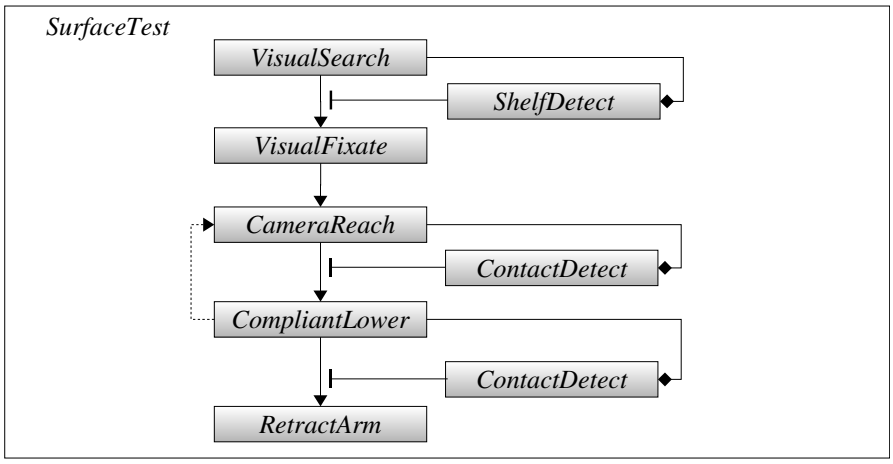


Figure 6-5: Much like a person reaching in the dark, the *SurfaceTest* module uses the robot's body to verify the presence of a surface. *ShelfDetect* finds a shelf edge that is then centered in the image by *VisualFixate*. The edge depth is unknown, so *CameraReach* extends the arm along the optical axis of the camera, expecting to bring the forearm in contact with the edge. If *ContactDetect* signals premature contact, the reaching posture is first adjusted. Otherwise, *CompliantLower* applies brings the hand to rest on the surface. If *ContactDetect* signals success, the successful arm posture is recorded and the arm retracted.

controller is forming a grasp. Second, if the net angular velocity of the fingers is close to zero, it is assumed that the fingers are in a stable state. Third, if $G_a(\mathbf{\Theta}) > 20mm$, then it is assumed that fingers are wrapped around an object and not resting on the palm of the hand. If all three conditions are true, then *GraspDetect* signals a stable grasp.

6.4 *SurfaceTest*

In unstructured environments, it can be difficult to use vision alone to determine with certainty the existence of an important feature. However, a robot can use its body to actively test a hypothetical feature location to learn more. In this vein, *SurfaceTest* uses the robot's arm to verify the existence and location of a reachable surface. People exhibit a similar behavior when placing an object on a surface in the dark. They will often first reach out and touch the surface. Doing this confirms their hypothesis of where it is and reduces their perceptual uncertainty.

Our implementation of *SurfaceTest* is specialized for a shelf surface that has a visible leading edge, though we are not necessarily restricted to these surfaces. The shelf edge is a common, task relevant feature in domestic settings. By only detecting the edge of a shelf, we mitigate the need for a 3D shelf model. *SurfaceTest* is illustrated in Figure 6-5. The algorithm is summarized as follows:

1. Visually identify a candidate shelf edge.
2. Reach out along a ray from the camera to a fixed depth above the shelf edge.
3. Adjust the reach if premature contact is made with the shelf.
4. Use compliant force control to move the hand down and make contact with the surface.
5. Detect contact (or lack of) with the shelf.
6. Store the posture, prior to descent, that led to success. Bring the arm back down to the side.

To begin, *SurfaceTest* identifies a shelf edge through modules *VisualSearch* and *ShelfDetect*. *VisualSearch* causes the robot's gaze to scan a room until *ShelfDetect* is signaled. *ShelfDetect* uses a HSV color filter and blob detector to detect two green stickers marking the edge of a shelf. This simple detector is not especially robust as there are often other green objects in Domo's office environment. Also, the appearant color of objects changes as the lighting changes throughout the day. Fortunately, *SurfaceTest* acts to reduce this perceptual uncertainty. *ShelfDetect* could be readily replaced by a more sophisticated technique using stereo information or surface texture, allowing *SurfaceTest* to work on a variety of surfaces without markers.

Once a shelf edge has been identified, *CameraReach* extends the arm along the optical axis of the camera. The module first defines an image target at a fixed height above the edge midpoint, corresponding to a point $\mathbf{x}^{ses}$ in the ego-sphere. Using *VisualFixate*, the head servos $\mathbf{x}^{ses}$ into the center of the image. Once its target is centered, the camera is held fixed so that visual occlusions of the shelf do not effect the arm controller. The depth of the shelf edge is unknown, so *CameraReach* controls the arm in depth along the camera's optical axis until contact is made. First, the arm closest to $\mathbf{x}^{ses}$ is selected and its hand is brought to a defined location ${\mathbf{x}^{\mathbf{c}}}_{start} = [0, 0, z_{start}]^T$ in the camera frame $\{C\}$. Using

the inverse kinematic model, the hand is extended along the camera's optical axis towards $\mathbf{x}^c_{end} = [0, 0, z_{end}]^T$.

We choose z_{start} and z_{end} so the arm starts close in to the body and ends at an arms length from the head. Typically, the reach stops short of $\mathbf{x}^c_{end}$ due to contact between the shelf edge and the forearm. The arm impedance is kept low during the arm extension, allowing it to safely stop short its target. If *ContactDetect* is signaled during extension, the arm posture is adjusted by retracting a fixed distance from the contact posture. This positions the hand just above the front edge of the shelf when *CameraReach* finishes.

Next, the *CompliantLowe*r module brings the hand to rest on the surface. It uses the gravity compensated force controller to generate a downward force f_z at the hand such that $\tau_{desired} = \mathbf{J^T}[0, 0, -f_z, 0, 0, 0]^T$. If a large hand displacement is detected, *SurfaceTest* assumes that either the shelf is out of reach or the feature detection was erroneous. Otherwise, *ContactDetect* detects the surface contact and the joint posture of the arm prior to *CompliantLower* is recorded. In this way, *SurfaceTest's* internal model of a surface is simply the joint angles that bring the hand above the surface. Other modules can now use this direct representation to easily control the arm relative to the surface. Finally, *RetractArm* brings the arm back to the robot's side.

6.4.1 Results

We tested *SurfaceTest* for 15 consecutive trials. For each trial the shelf was moved to an arbitrary location near the robot. The shelf height was varied between $0.05m$ and $-0.25m$ relative to the shoulder. Some locations were deliberately out of reach of the robot, and some were too close in. A trial was successful if the robot correctly verfied that the shelf edge could be reached, and if so, the hand came to rest on the front edge of the shelf. The module was successful for all 15 trials, and the results of five of the trials are shown in Figure 6-7. The robot is shown executing the module in Figure 6-6. Although the algorithm is shown to be robust to the shelf location and height, it the arm can get trapped under the shelf if it is too close to the body. In this case, coarse knowledge of the surface depth would be useful in guiding *CameraReach*.

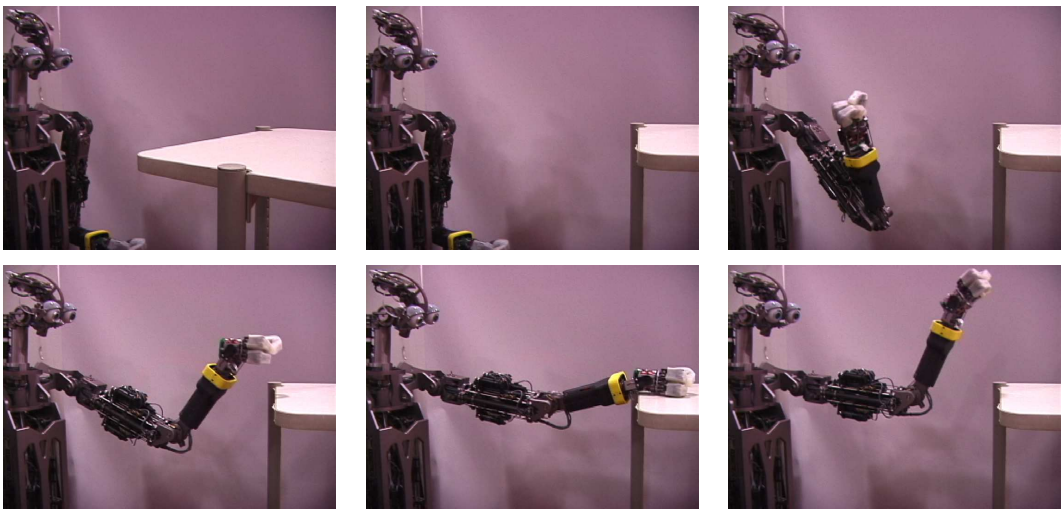


Figure 6-6: Domo's *SurfaceTest* behavior verifies the presence and location of a shelf. In this demonstration, the shelf location is first moved and then *SurfaceTest* reestimates the location.

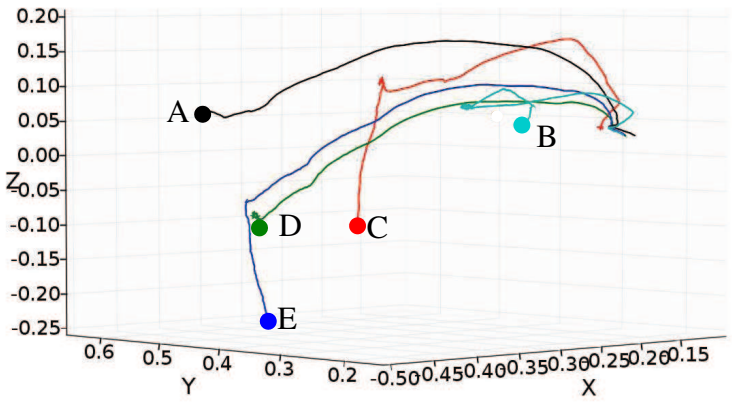


Figure 6-7: The reaching trajectory, in meters, of the hand during five consecutive trials of *SurfaceTest*. The final pose of the hand is annotated. The shelf surface is moved to five arbitrary locations in the robot's workspace and the shelf height varied. In trials A,B,and C, the shelf height is at .05m relative to the shoulder. In trials D,E the height is at $-.15m$. For trials A, B, and D, Domo successfully rests its hand on the front edge of the shelf. In trials C and E, Domo correctly detects that the shelf is out of reach. For Trial B, the shelf is much closer in and the arm contacts the shelf prematurely. This is detected and the posture adjusted to successfuly find the shelf edge.

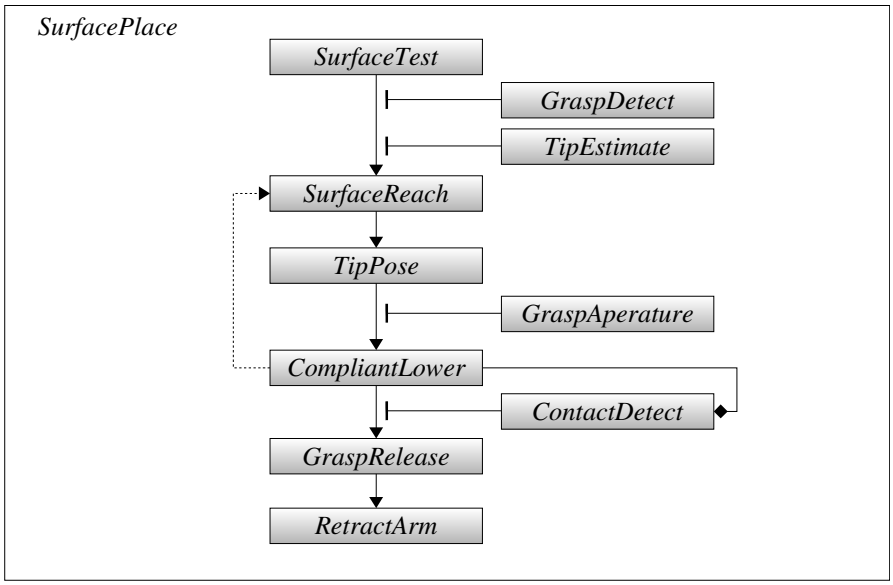


Figure 6-8: The *SurfacePlace* module transfers an object from the robot's hand to a surface. First, *SurfaceTest* identifies a location to place objects. *TipEstimate* finds an objects alignment axis such that *TipPose* can align the object to the surface. Depending on *GraspAperture*'s estimate of the object size, the object is aligned horizontally or vertically to the surface. *SurfaceReach* brings the object over the surface and *CompliantLower* allows it to reorient to the surface as it is brought into contact. Finally, when *ContactDetect* is signaled, *GraspRelease* opens the hand and the arm is retracted.

6.5 *SurfacePlace*

Human environments are dominated by flat surfaces, and many useful tasks involve placing objects on surfaces. Stocking goods, setting the table, arranging a product display, placing a part onto a conveyor belt, and putting dishes are just a few of the everyday tasks that require surface placement. Fortunately, many man-made objects are designed so their intrinsic mechanics assist placement in a canonical orientation. For example, a wine glass has a wide base allowing it to remain upright on a shelf. We would expect a pencil to rest on its side. There are exceptions of course. Books tend to lay on their sides, but convention dictates that it rests upright on a bookshelf.

In this vein, the *SurfacePlace* module takes a grasped object and places it on a nearby flat surface in one of two canonical orientations: upright or lying down. If the object's canonical orientation is upright, we assume that the object has a sufficiently large flat base to rest on. The module is illustrated in Figure 6-8, and its algorithm can be summarized as follows:

1. Detect a useable surface.
2. Find the alignment axis of a grasped object.
3. Determine if the object is to be placed upright or lying down.
4. Reach to above the surface.
5. Align the object to the surface.
6. Decrease the manipulator impedance and use force control to lower the object onto the surface. Allow the object to self-align using compliance in the hand and wrist.
7. Release and retract.

This algorithm requires a few assumptions about the object in order to align its base to the surface. First, the aligment axis, by our definition, brings the object's flat base parallel to the surface when it points in the direction of gravity. For unknown objects, we can treat the alignment axis as a task relevant feature that requires perceptual estimation. We simplify this problem by assuming that when an object is grasped, its alignment axis extends from the center of robot's palm to the distal tip of the object. This assumption is true for a

variety of everyday objects such as bottles, books, and many hand tools. For objects that this assumption holds, we can visually detect and control the alignment axis using the T*ipEstimate, TipPriors,* and *TipPose* modules. These modules are described in Chapter 8.

Once the aligment axis is detected and *SurfaceTest* has located a useable surface, *SurfaceReach* positions the arm just above the shelf surface. Depending on the grasp aperture, the aligment axis is posed either normal or parallel to the surface. This assumes that the size of the object's base can be estimated *GraspAperture.* Objects with grasp apertures below 30mm are placed lying down on the surface because they would likely fall over if placed upright. Once the object has been aligned, *CompliantLower* lowers the arm stiffness and brings the object down onto the shelf surface using force control. When *ContactDetect* is signaled, the grasp on the object is released and the arm retracted.

An important aspect of *SurfacePlace* is that it exploits compliance in the wrist and hand, as well as contact with the surface, to align the object's base to the surface. In this way, Domo can achieve a goal orientation despite uncertainty in the pose of the grasped object. This strategy has a long history in manipulation, where the intrinsic mechanics of an object in contact with the environment are used to reduce uncertainty. Notable examples include Mason's analysis of manipulation funnels [89], Inoue's use of force sensing for peg-in-hole tasks [66], and the development of the remote-center-compliance (RCC) wrist for passive alignment of objects [141].

6.5.1 Results

SurfacePlace relies upon the manipulator compliance to assist in realignment of an object while placing it. As shown in Figure 6-9, we measured the success of *SurfacePlace* in placing a bottle upright on a shelf as the wrist stiffness, K_{ps}, and misalignment angle, θ_s, are varied. We define θ_s as the angle between the surface plane and the object's base. As the figure illustrates, in the ideal case the expected misalignment tolerance is $\theta_s < \frac{\pi}{2} - tan^{-1}\frac{h}{r}$ for base radius r and grasp height h. In this experiment, we expect *SurfacePlace* to be successful when when $|\theta_s| < 21.2^\circ$ where $r = 35mm$ and $h \approx 90mm$.

In the experiment, the shelf location was fixed and the bottle was repeatedly handed to the robot in a near identical pose. When grasped, the bottle's base is approximately parallel to the bottom of the hand, allowing for θ_s to be measured kinematically. We tested $-40^\circ \leq \theta_s \leq 40^\circ$ in 5 degree increments and wrist stiffness $K_{ps} \in [0.0, 0.5, 1.0]$. We

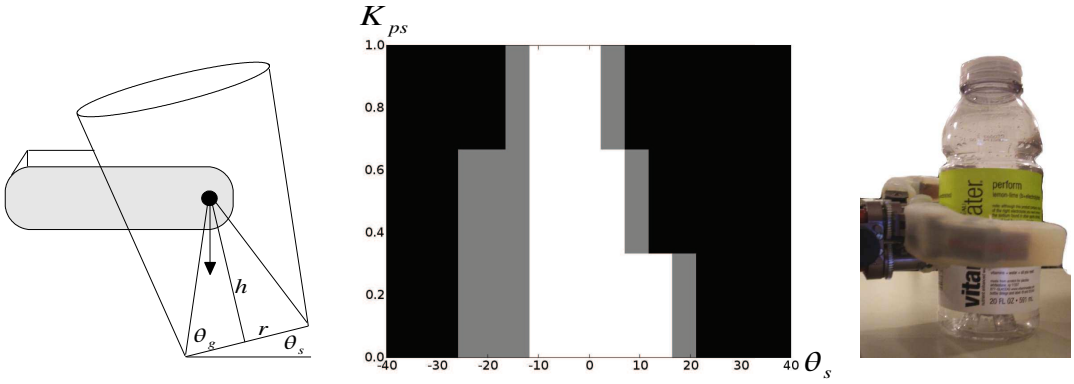


Figure 6-9: *SurfacePlace* uses contact with a surface to passively align the base of an object to the surface. **Left**: A downward force applied to the object creates a realignment moment about the contact point. Ignoring the object's mass and friction, a lower grasp or a wider base will increase the robustness of this strategy. We can expect success when $\theta_s + \theta_g < \frac{\pi}{2}$ for $\theta_g = tan^{-1}\frac{h}{r}$. **Middle**: A 2D histogram showing the probability of success in placing a bottle upright on a shelf as the wrist stiffness, K_{ps}, and the misalignment angle, θ_s, are varied. As the stiffness is lowered, the toleance to misalignment improves. **Right**: The bottle used in this experiment.

Figure 6-10: A variety of everyday objects were successfully tested with the *SurfacePlace* behavior, including (left-right): stuffed animal, shoe box, large water bottle, headphone caddy, spoon, paint roller, hand broom, food tin, small water bottle, duster, and a spray bottle.

conducted two trials of each of the 51 parameter combinations. The force at the hand was approximately $[0, 0, -4]^T$ Newtons. A trial was successful if the bottle was left standing upright on the shelf. The experiment results are shown in Figure 6-9 as a 2D histogram. They demonstrate that the misalignment tolerance improves as the wrist stiffness is lowered. For $K_{ps} = 0$, the tolerance is roughly ± 20 degrees as predicted. In practice, many other factors complicate the success of *SurfacePlace.* Unstable objects can be disturbed as the hand is withdrawn. The wrist does not achieve true zero impedance, and compliance in the hand introduces additional adaptation of the object's alignment.

Next, we tested *SurfacePlace* on a wide range of everyday objects. Some of these objects are shown in Figure 6-10, including a stuffed animal, shoe box, large water bottle, headphone caddy, spoon, paint roller, hand broom, food tin, small water bottle, duster, and a spray bottle. The broom, duster, and spoon were correctly placed on their sides due to their small diameter handles. The shoe box and headphone caddy are large and non-cylindrical, violating the assumptions of *SurfacePlace*. However, they were still smoothly lowered onto a stable base. The paint roller was the most difficult to consistently place upright due to its small, $40mm$ diameter base.

6.6 Discussion

In this chapter we have demonstrated that a robot can leverage its physical embodiment to assist in perception and to reduce uncertainty. It illustrates the design theme of *let the body do the thinking.* In particular, we have emphasized algorithms that do not require precise geometric and kinematic knowledge about the state of the robot and the world. For example, the robot can leverage low manipulator impedance and passive compliance to detect unexpected contact with the world or to allow a grasped object to reorient during placement. The *SurfaceTest* module demonstrates a compensatory behavior, where the robot takes action to test a perceptual hypothesis. We should also note that, as much as possible, modules such as *GraspDetect* use the sensory state of the robot in the world rather than an internal representation of its state. In our experience, this has increased the robustness and responsiveness of the robot, especially during unexpected task failures that are often left out of internal models.

The modules presented in this chapter could certainly be expanded in several ways.

Sensing of the contact moments during *SurfacePlace* would allow Domo to actively adjust the object's orientation. *SurfaceTest* could be expanded beyond shelf edges to include any type of flat surface. Limited force sensing resolution and the simple model used in *ContactDetect* prevents Domo from knowing the wrench of contact forces acting on the manipulator. This information could be useful in developing more adaptive algorithms. Finally, *StiffnessAdapt* simply sets the manipulator impedance to a hand designed value at a modules request. It would be profitable for this impedance to be autonmously adapted in response to sensory feedback during the task.

Chapter 7

Cooperative Manipulation

In this chapter we investigate key aspects of the cooperative manipulation design strategy. We present the *PersonDetect* and *VocalRequest* perceptual modules that enable Domo's real-time interaction with people. We then describe the *AssistedGrasp* module which cues a person for assistance in grasping an object. Finally, we test experimentally the hypothesis that people will intuitively assist Domo in a manipulation task without prior instruction.

7.1 Perception of People

We would expect a robot collaborator to be responsive to our presence in a room, to understand the intent of our gestures, and to adapt to our feedback. Ultimately, cooperative manipulation requires the robot to understand referential (looking and pointing) and goal-directed (reaching) cues. Clearly, this is an open area of research involving many difficult perception problems such as gesture recognition and learning from demonstration. However, many cooperative manipulation tasks can be designed around much simpler communicative cues so long as the cues are robustly detected and a person can generate them without much effort. For example, in the previous chapter we saw that *ContactDetect* can be used to detect a person grabbing the arm as it moves. A person will intuitively generate this cue to provide negative feedback to a collaborator. In this section we present modules that allow Domo to detect and track a collaborator, react to a hand waving cue, and respond to vocal requests. In developing these real-time visual algorithms, we have emphasized robustness to cluttered, everyday environments with variable lighting.

Figure 7-1: During human-robot interaction, hands, fingers, and faces are important perceptual features for visual attention. As shown in green, the *InterestRegions* module will often selects these features because they are well modeled as rapidly moving convex edges.

7.1.1 Detecting Hands, Fingers, and Faces

During a cooperative manipulation task, much of a person's attention is directed towards the hands, fingers, and face of their partner as well their own hands and fingers. Detecting these features can be a difficult task for a robot working in a dynamic and unstructured environment. However, in collaboration with Kemp [76], we have found that the *InterestRegions* module will often select for these features during human-robot interaction.

The interest point operator used by *InterestRegions* detects convex edges that are moving rapidly with respect to the background. A hand will often produce the fastest moving edge in the image because it is the furthest point from the arm's center of rotation. It also has a roughly convex projection in the image. The same holds for fingertips and heads. During social interactions, people naturally gesture with these features and bring them close to the camera. The motion from a gesture can serve as a natural cue to direct the robot's attention. This is illustrated in Figures 7-1 and 5-4.

We evaluated the ability of *InterestRegions* to select for these features during human-robot interaction. Prior to the experiments we manually moved the arm around its reachable workspace and collected 4000 samples of arm postures that could be safely commanded to the motor controllers. During an experiment, the robot grasped an object and periodically $(0.25Hz)$ executed a reaching movement to a random arm posture generated from the collected data. As the arm explored its workspace, a person's interaction with the robot involved full-body motion, hand waving, presentation of objects, as well as physical contact with the robot's arm as it moved. We conducted two interaction experiments, each lasting

Figure 7-2: A random sample of image patches collected by the *InterestRegions* module. These predominantly contain hands, fingers, and faces.

about 2 minutes and generating approximately 2000 samples.

For each sample we used the *InterestRegions* module to select the most salient region in the image. An image patch was selected based on the scale and location of the interest region. Figure 7-2 shows a random sample of image patches. The majority of the patches correspond to features relevant to human-robot interaction, including the person's head, eyes, hands, and fingers, the objects being presented, and the robot's hand and fingers. We hand categorized 200 randomly selected image patches from the data set. Figure 7-3 shows that over 50% of these patches were of human features, and over 30% were of the robot's hand. Although these experiments were conducted while keeping the robot's gaze stationary, similar results have been obtained when the head is allowed to move.

7.1.2 *PersonDetect*

PersonDetect is a module in the visual attention system dedicated to perception of the robot's cooperative partner. As illustrated in Figure 7-4, it detects and tracks a person's face, models the skin color of the face, and uses the skin color and *InterestRegions* module to detect a person's waving gesture.

Tracking Faces

We use the Viola-Jones face detector provided in OpenCV [34, 138] to detect one or more faces in $160x120$ monochrome images at $30hz$. Whenever a face is detected, a $4x4$ pixel

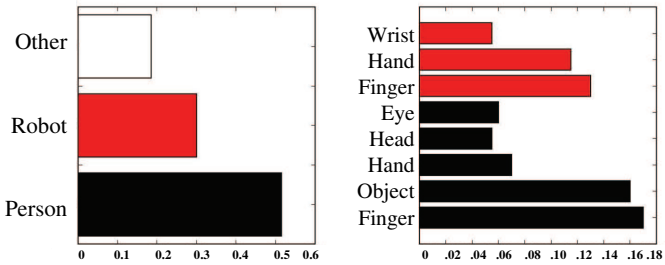


Figure 7-3: Statistics of feature categories detected during human-robt interaction. We hand-labelled categories for 200 image patches randomly sampled from the image patches collected by the visual attention system. A patch was labelled as a person (black bars) if it selected either a hand, finger, head, eye, or object in the hand. A patch was labelled as a robot (red bars) if it selected either the robot's hand, finger, or wrist. Patches that were neither person nor robot were labelled as other. The left plot shows the probability of each category and the right plot shows the probability of each sub-category.

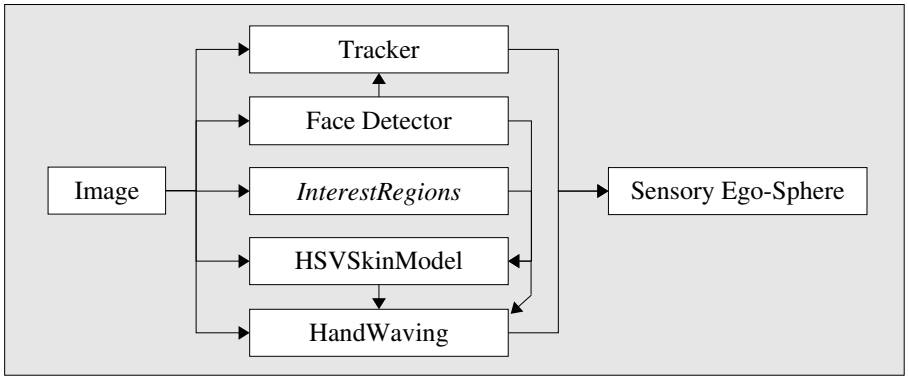


Figure 7-4: The *PersonDetect* module. A face detector initializes a tracker and the tracked face is used to build an online model of the person's skin color. This model is combined with *InterestRegions* to select for salient moving features on a person. These often correspond to the person's hand. Accumulated evidence of hand motion is detected as a hand waving cue. The face and hand waving features are made available to the SES in order to direct the robot's atttention.

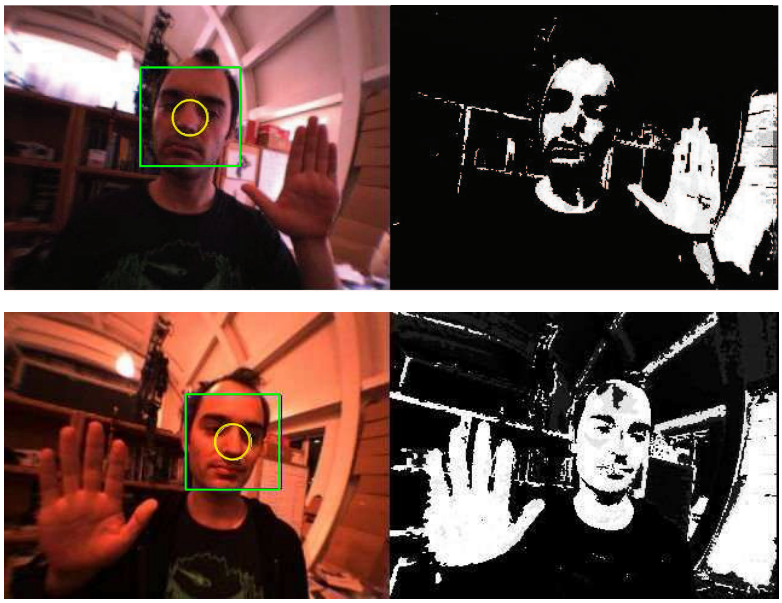


Figure 7-5: Output from the face tracker and skin color filter. The HSV skin color model adapts to the large shift in appearant skin hue between daylight (top) and night(bottom).

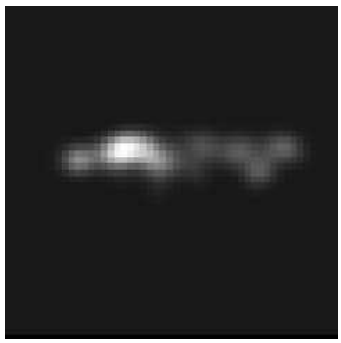


Figure 7-6: The 2D histogram representing the accumulated probability distribution, $p(\mathbf{x}^{ses}|face)$, for face detections within the SES. The center of the histogram, $\mathbf{x}^{ses} = [0, 0, r]$ corresponds to the robot looking straight ahead.

block centered on the face is used to initialize a tracker. The tracker uses block matching over a $6x6$ pixel window to track the block between consecutive images. The location of a tracked face is continually refreshed within the SES. The tracker also produces a confidence measure, defined as the pixel sum difference between the current block and the intialization block. The tracker times out and assumes it has lost the face when the confidence is below a threshold for longer than 60 frames. Fortunately, people usually move slowly within a scene, and the timeout allows the face detector time to reacquire a frontal view of the face. The output from the face tracker is shown in Figure 7-5.

The Spatial Distribution of Faces

As Domo interacts with people and tracks their faces, *PersonDetect* models the spatial distribution of the face detections within the SES. This model is learned over an extended period of many human-robot interactions. The detections are used to estimate the probability distribution $p(\mathbf{x}^{ses}|face)$ of Equation 5.5, which represents the chance of seeing a face at location $\mathbf{x}^{ses}$ within the SES. Figure 7-6 shows the distribution after many hours of interaction with the robot. Once learned, $p(\mathbf{x}^{ses}|face)$ can provide a confidence measure on a new face detection. For example, *PersonDetect* will ignore a face detection if $p(\mathbf{x}^{ses}|face) < 0.35$. This prevents the robot from falsely detecting faces on the floor or ceiling.

Modelling Skin Color

When a face is detected, the image patch within the face bounding box is used to automatically model the person's skin color. The patch is first converted from *RGB* to *HSV* color space. The hue and saturation of each pixel is added to a 2D $16x16$ bin histogram. This defines the probability distribution $p(\mathbf{x}|skin)$, which represents the chance that pixel $\mathbf{x}$ is skin colored. Now given a region $\mathbf{X} = [\mathbf{x}_1...\mathbf{x}_n]$ within the image, we compute the probability that the region is skin colored as the average of each pixel:

$$p(\mathbf{X}|skin) = \frac{1}{n}\sum_{\mathbf{x}\in\mathbf{X}} p(\mathbf{x}|skin).$$

Also, each contribution to the histogram is given an initial weight of 1.0. This weight is decremented by a small (.001) amount each time step, biasing the histogram toward more recent face detections. As shown in Figure 7-5, this allows the skin color model to adapt to

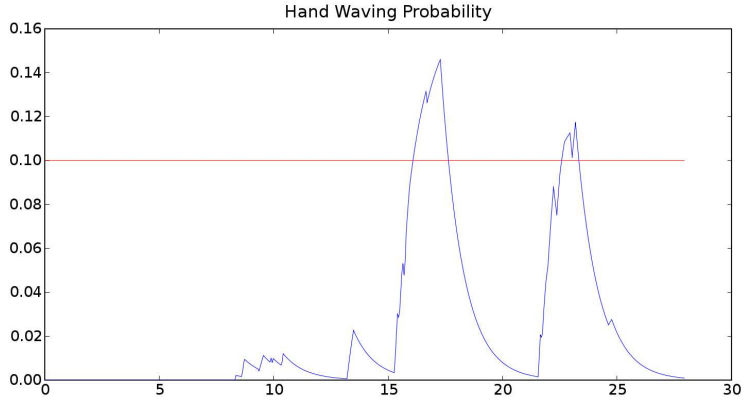


Figure 7-7: The probability of hand waving, $p(\widehat{\mathbf{x}^{ses}}|hand)$, over a 30 second trial. The detector threshold $\epsilon_h = 0.1$ is shown in red. The skin color model is built during the first 10 seconds of the trial. Head motion appears during time $10s - 15s$ with a low probability. Two brief hand waving episodes are detected in the last 15 seconds of the trial.

new people and changing lighting conditions. This approach is similar to that used on the ARMAR robot [124].

Waving Detection

PersonDetect also detects hand waving. As we demonstrated previously, features selected by *InterestRegions* often correspond to hands and faces. An interest region $\mathbf{X}$ in the image corresponding to location $\mathbf{x}^{ses}$ in the SES is labelled as a hand feature whenever the region is skin colored, it is not a face, and a face is present in the SES. In other words,

$$p(\mathbf{X}|skin) > \epsilon_s,$$

and

$$p(\mathbf{x}^{ses}|face) < \epsilon_f$$

, for thresholds ϵ_s and ϵ_f. This simple hand detector is prone to false positives such as moving flesh colored objects and other moving body parts. However, false detections are generally distributed across the SES. Repeated hand motion in one location can accumulate evidence over time that a person is waving to the robot. Evidence is accumulated by modelling the spatial distribution of hand detections in the SES as $p(\mathbf{x}^{ses}|hand)$ using Equation

5.5 with $\Delta w_{hand} = 0.1$. Finally, *PersonDetect* selects the maximum likelihood location, $\widehat{\mathbf{x}^{ses}}$, of the distribution. *PersonDetect* signals hand waving whenever $p(\widehat{\mathbf{x}^{ses}}|hand) > \epsilon_h$ for some threshold ϵ_h. In Figure 7-7 we plot $p(\widehat{\mathbf{x}^{ses}}|hand)$ as Domo interacts with a person. We see that the hand waving is readily detected.

7.1.3 *PersonSeek*

The *PersonSeek* module allows the robot's gaze to be responsive to a person in the room. Its algorithm is straightforward:

1. If a face is present in the SES, servo the head to track the face as they move around the room using *VisualFixate*.

2. If a face is not present in the SES, periodically ($0.25Hz$) redirect the robot's gaze to locations that have a high probability of finding a person. A gaze location $\mathbf{x}^{ses}$ is drawn from the distribution $p(\mathbf{x}^{ses}|face)$.

3. Whenever hand waving is detected, or $p(\widehat{\mathbf{x}^{ses}}|hand) > \epsilon_h$, redirect the gaze to location $\widehat{\mathbf{x}^{ses}}$.

The module gives priority to hand waving over face detection. This allows a person to guide the robot's attention away from the face to an object of interest. Once the robot is attending to the object, other modules can assume control of the head if desired. Otherwise, the robot's gaze is returned to the face after a 5 second timeout. In the absence of any competing modules seeking control of the robot's head, *PersonSeek* will be active. For example, after a manual task is complete, the *PersonSeek* automatically redirects the gaze to the person. This provides an intuitive cue to the person that the robot has finished its task.

7.1.4 *VocalRequest*

Domo has a simple speech interface, allowing it to respond to and generate vocal requests. Motor noise typically degrades voice recognition performance when using microphones attached to the robot's body. Consequently, we require that a person talking to Domo wear a wireless Sennheiser lavalier microphone. The *VocalRequest* module uses the CMU Sphinx2 voice recognition package freely available on Linux. Sphinx is configured to recognize a

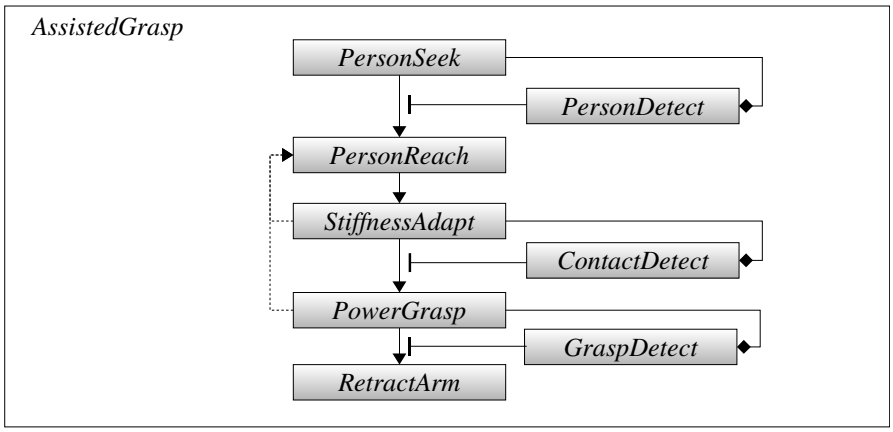


Figure 7-8: The *AssistedGrasp* module cues a person for assistance in grasping an object. When *PersonDetect* signals the presence of a person, *PersonReach* extends the nearest arm to the person with an open hand. If the cue is interpreted correctly, the person places an appropriate object in the robot's hand. Low manipulator impedance allows *ContactDetect* to sense contact as the object is handed. This triggers *PowerGrasp* to secure a grasp on the object. If the contact is not detected, or a stable grasp isn't formed, then the robot re-cues the person to try again. Otherwise, the arm is retracted holding the grasped object.

small vocabulary of commands such as "Take", "Give", and "Shelf". The prefix "Domo" is required, allowing *VocalRequest* to ignore speech that is not directed at it. In response to a command, Domo repeats the phrase using the Digital DecTalk voice synthesizer, allowing the person to confirm that they were heard correctly. *VocalRequest* then increases the dynamic priority of the command's module. For example, saying "Domo, put this on the shelf" will increase the priority of the *SurfacePlace* module.

7.2 *AssistedGrasp* and *AssistedGive*

With the *AssistedGrasp* module, Domo is able to gain a person's help in grasping an object. This is a simple yet important example of cooperative manipulation. The robot, aware of its collaborator's presence, extends an open hand towards the person. If this gesture is generated in the appropriate context, it should be understood by the collaborator as a request to be handed a particular object. If Domo is successful in enlisting the collaborator's assistance, Domo will detect the object being placed in its hand and form a stable grasp on

Figure 7-9: Domo executing the *AssistedGrasp* module to grasp a water bottle and spray bottle. Domo visually detects the collaborator and reaches towards him with an open hand. Domo detects the objects being placed in its hands and forms a power grasp on them.

it. The *AssistedGrasp* module is illustrated in Figure 7-8 and its algorithm is as follows:

1. Look for and detect a person.
2. Reach to the person with an open hand while directing the eye gaze to their face.
3. Lower the arm stiffness.
4. Detect the contact forces as the object is placed in the hand.
5. Form a power grasp and detect its success.
6. If a successful grasp isn't made, re-cue the person. Otherwise, retract the arm.

AssistedGrasp can be activated so long as one hand is empty. First, it employs *PersonSeek* to find a person in the room. If *PersonDetect* finds a face at location $\mathbf{x}^{ses} = [\theta, \phi, r]$, then *PersonReach* uses the inverse kinematic model to reach to the target $[\theta, \phi + \frac{\pi}{4}, r]$. This location should be near the person's midriff. The orientation of the extended hand and the pose of its fingers can also be used to cue the person how to hand the object. For example, a glass of water is usually grasped in a dfferent orientation than a stirring spoon. This desired orientation can be specified by an external module.

Once *PersonlReach* has achieved its target and the arm is nearly stationary, *StiffnessAdapt* lowers the stiffness of the arm. This increases the likelihood of *ContactDetect* given small contact forces. As the person gives Domo the object, small displacements of the hand are sensed by *ContactDetect*. *PowerGrasp* then closes the fingers around the object. If *GraspDetect* signals success, *RetractArm* brings the grasped object to the robot's side. However, if *ContactDetect* or *GraspDetect* fail, then *PersonReach* is reactivated and the robot cues the person again. Figure 7-9 shows the execution of *AssistedGrasp*.

AssistedGrasp is typically activated by other modules that require assistance in grasping an object. However, *AssistedGrasp* can also assist the collaborator by simply holding something for them. For example, a person can say "Domo, hold this" and *VocalRequest* will activate *AssistedGrasp*, causing Domo to reach out and take the object. In another scenario, a person can non-verbally request assistance by simply waving the object in front of Domo. The hand motion is detected by *PersonDetect* which will also activate *AssistedGrasp*. In these ways, *AssistedGrasp* allows for natural interaction between the robot and collaborator.

While *AssistedGrasp* takes an object from a person, the *AssistedGive* module hands a grasped object back. Its implementation is nearly identical to *AssistedGrasp,* only now *GraspRelease* is used instead of *PowerGrasp.* If a collaborator had asked for assistance by requesting "Domo, hold this", they can now request "Domo, give it" and *AssistedGive* will cause Domo to reach to the person and hand the object back.

7.3 Testing Cooperative Manipulation

If we take *cooperative manipulation* as a design strategy, to what extent can we include the actions of a collaborator into the task design? This is an important question if we wish to develop manipulation strategies that depend on people for task success. By placing the collaborator "in the loop", a robot can do more with less. However, this should also be a beneficial experience for the person, otherwise they won't be inclined to work with the robot. Not only should the robot do something useful, but the collaboration should occur in a natural, intuitive manner without requiring excessive training for the collaborator or mental effort in assisting the robot.

One way to induce such a collaboration is to have the robot to use socially understood gestures and cues. We maintain that such cues will be interpreted correctly if they are generated in the appropriate context. For example, *AssistedGrasp* expects that a collaborator will understand the reaching cue as a request for a particular object. This assumes that the collaborator will assess the task context to select the correct object. People exhibit this behavior when they pass their coffee cup but not their dinner plate to a waitress holding a pot of coffee.

We also maintain that a collaborator will develop an understanding of the robot's perceptual and motor limitations. A well intentioned collaborator will subsequently adapt their actions to increase task success. As we will see, during *AssistedGrasp,* a collaborator will intuitively hand the object to the robot in a pose that anticipates both the robot's use of the object and the limitations of its grasping.

This type of cooperative interaction as been investigated recently with the Dexter humanoid [55]. Dexter learns that when a desired object is out of reach, the act of reaching towards the object will induce a person to move the object closer. However, the person's cooperation occurrs as part of the experiment's design and not because they are compelled

to. Can a robot compell a person to take on the role of collaborator without instruction? In this section we present an experiment that investigates this question.

7.3.1 Give and Take Experiment

Experiment Setup

As shown in Figure xxx, the subject sits to the side of the robot. The robot is at a table and an oblong box (60x85x200mm) sits on the table. The box is instrumented with a gyroscope to measure its orientation. The subject is told only that the robot is performing an unspecified visual hand-eye calibration task, and that whenever the robot reaches to her, she is to place the box in the robot's hand. This explanation is to deter her from explicitly considering the way in which she hands an object to the robot. In a single trial, the robot reaches to the subject using *AssistedGrasp* with the hand open in a power-grasp preshape. The subject places the box in the robot's hand and the robot grasps the box, brings the box up to its cameras, and appears to inspect it. Depending on the subject, it then lowers its arm in one of two ways.

In the first case, the robot reaches towards the person, bringing the box just in front of and above the table edge nearest the subject. It says the word "Done" and pauses for one second. It then releases its grasp, dropping the box onto the table, and retracts its arm. In the second case, the robot does an identical action as in the first case, but this time it reaches just past the table edge. Unless the person takes the box from the person, it falls to the floor. This marks the end of a trial. The robot pauses for 5 seconds and then initiates the next trial. Six trials are performed with each subject.

At the start of each experiment, the box is aligned to the robot's body and the gyroscope is calibrated with respect to frame $\{W\}$. We define the vector $\mathbf{b}^w$ as the longest edge of the box. We define the power-grasp axis as $\mathbf{z}^h$ which is the z-axis of frame $\{H\}$ in Figure 4-1. This axis corresponds to the long axis of a soda can held in a power grasp. In frame $\{W\}$ this axis is $\mathbf{z}^w$. The angle between $\mathbf{z}^w$and $\mathbf{b}^w$ is defined as the grasp aligment error. During each trial, we measure the average grasp alignment error during the $500ms$ just prior to the grasp being formed. We also vary the wrist rotation for each of the six trials such that the angle formed between $\mathbf{z}^w$ and gravity varies by $[0, -45, 90, 0, -45, 90]$ degrees.

Experiment Hypothesis

This experiment considers the following three questions:

1. When a subject hands the robot the box, does she adjust its orientation to match the pose of the robot's hand?

2. Will the subject correctly interpret the robot's reaching gesture, vocalization, and pause as a social cue to take the object?

3. Can a small incentive such as not having to pick up the object increase the subject's willingness to respond to the social cue?

We can use the measured grasp aligment error to directly answer the first question. We would expect to see $\mathbf{b}^w$ to track $\mathbf{z}^w$ as it varies between the three wrist orientations. The second queston is more difficult to confirm. For each experiment, we measure the take-back rate as the percentage of trials a subject reached to take the box back from the robot. We expect that robot's reaching gesture, vocalization, and pause will cause the subjects to take back the box. However, they are never instructed to take it back, so if they do, then it is likely that they correctly interpreted the cue, much as they would with a person. For 50% of the subjects, the robot drops the cylinder on the floor. If this serves as an incentive, we should see an increase in the take-back rate.

Experiment Results

Prior to the experiments, we first measured the average grasp aligment error when the box was deliberately oriented to match the robot's grasp. From repeated trials the mean error was 7.6 degrees. Next, we measured the range of grasp aligment errors when the box was freely rotated within the preshaped hand. In this case the distribution of grasp errors is uniform between 0 to 60 degrees.

We tested the experiment using 15 subjects as well as ourselves for comparison. After an experiment each subject was asked to rate, on a scale of 1-4, their level of experience working with robots, the amount of thought put into how the box was handed to the robot, and the level of expectation experienced to take the box from the robot. As shown in Figure 7-10, a typical subject matched the box orientation to the grasp orientation as predicted. The full

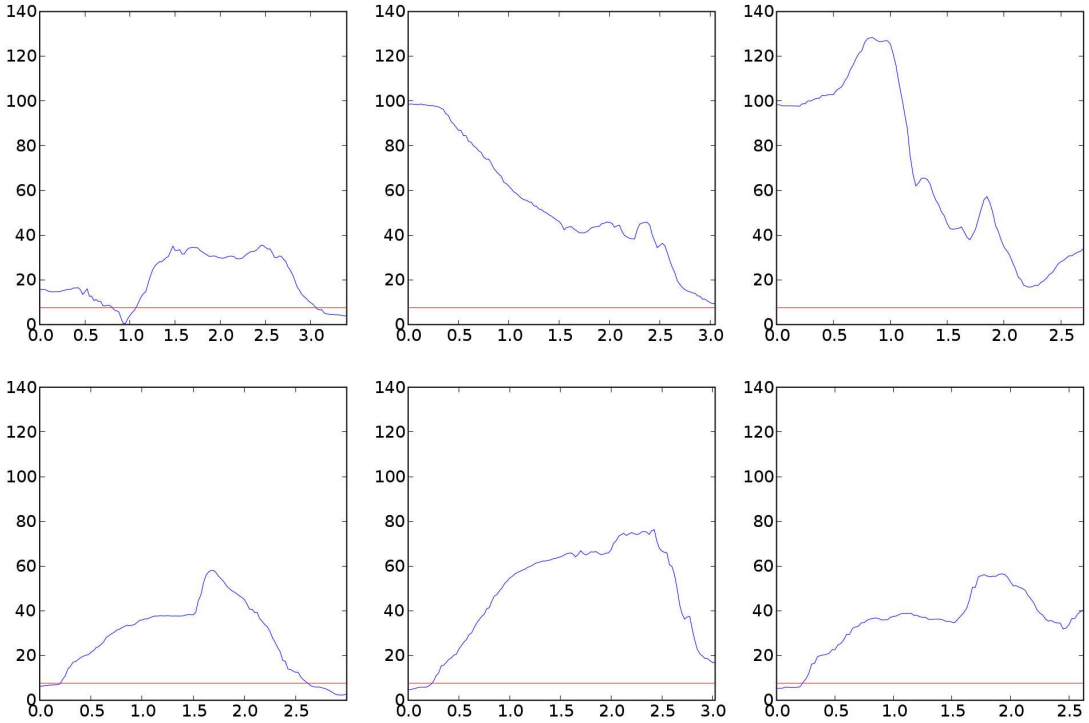


Figure 7-10: The grasp alignment errors for a typical subject during the Give and Take experiment. Blue shows the grasp alignment error (degrees) of the box with respect to the grasp preshape. Red shows the mean error achieved when we deliberately aligned the box with the grasp. The X axis shows the trial time, in seconds, starting when the reach commenced and ending when the grasp was initiated. We see that for nearly all trials, the subject aligns the box within a few degrees of the best expected performance. Interestingly, in the last trial the subject reported that they were deliberately testing the robot's ability to grasp a poorly aligned box.

Subject/Trial	Experience	Intent	1	2	3	4	5	6	Avg. Error
Author	4	4	4.4	8.6	5.7	3.1	2.3	0.5	4.1
1	4	2	1.7	14.7	2.7	19.3	5.0	4.8	8.0
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									

Figure 7-11: Results showing that the 15 subjects intuitively aligned the cylinder's axis with the grasp axis of the robot's hand when handing it. After the experiment, each subject rated on a scale of $1-4$ their level of robotics experience and the level of intent given to the object's placement. For each of 6 trials, we show the average alignment error, in degrees, during the 500 ms prior to grasping. The last column shows the average error for each subject. We also show the author's

Subject/Trial	Expertise	Intent	Dropped	1	2	3	4	5	6	Rate
1	4	2	No	1	1	1	1	1	1	100%
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										

Figure 7-12: xxx

results are shown in Figure 7-11. We also found that most subjects correctly understood the robot's cue and took the box back. These results are shown in Figure 7-12.

This experiment only scratches the surface of the potentially rich interactions that may occur during cooperative manipulation. However, it shows quantitatively that people will intuitively adapt to and assist the robot without instruction. We would expect that more substantitive assistance could be given if the person possesed greater contextual knowledge about the task and the robot could generate more nuanced cues.

7.4 Discussion

Cooperative manipulation can be informed by work in human-robot interaction, social robotics, as well as cognitive psychology. For example, to asseses an infant's understanding of a person as intentional agent, researchers studies have consiered eye-gaze in the presence of a pointing and reaching adult as an indicator of their understanding of people as intentional agents. [121]

1. people bring their years of experience to the table for the robot's benefit
2. what experiment tells us about people working with robot: that can design the person into the loop
3. also human infant studies for reaching and eye gaze, hri citation

Chapter 8

Task Relevant Features

Everyday tasks such as placing a cup in a cabinet or pouring a glass of water can be described in terms of the perception and control of *task relevant features*. We have seen previously how the edge of a shelf can represent the presence of a stable, flat surface. In this chapter we show that the distal tips of many everyday objects can be treated as task relevant features. We first present a robust method to estimate the 3D location of an object's tip in the hand's coordinate frame. This work was developed in collaboration with Charles Kemp [74]. Next, we describe the *TipPose* and *TipServo* modules which provide manipulator control of this feature. We then generalize the visual detection of the tip to include a broader class of objects. This work was also done in collaboration with Kemp [75]. We then show that a prediction of the feature location can be learned. Finally, we integrate these components into the *TipUse* module which enables Domo to take an object from a person, quickly find its distal tip, and control the tip for a task.

8.1 The Task Relevant Tool Tip

For a wide variety of tools and tasks, control of the tool's endpoint is sufficient for its use. For example, use of a screwdriver requires precise control of the tool blade relative to a screw head but depends little on the details of the tool handle and shaft. Radwin [110] describes 19 categories of common power and hand tools. Approximately 13 of these tool types feature a distal point on the tool which can be considered the primary interface between the tool and the world.

The prevalence of this type of feature may relate to the advantages it gives for perception

A B C

Figure 8-1: [A] A typical view from the robot's camera of the hand holding a pair of pliers. A naturally lit, cluttered background ensures a non-trivial unstructured environment for perception. [B] The tool-tip is detected as the maximum of the motion estimator's weighted edge map. [C] The raw motion-based detection (black), the hand-labeled tool tip (white), and a prediction based on the estimated tool position (green) are annotated.

and control. For perception, it improves visual observation of the tool's use by reducing occlusion, and it assists force sensing by constraining the interaction forces to a small region. For control, its distal location increases maneuverability by reducing the possibility of collisions. A single tip also defines the tool's interface to the world as a simple, salient region. This allows the user to perceptually attend to and control a single artifact, which reduces cognitive load. Looking beyond human tools, one can also find this structure in the hand relative to the arm, and the finger tip relative to the finger.

In this section, we present a straight-forward monocular method for rapidly detecting the endpoint of an unmodeled tool and estimating its position with respect to the robot's hand. The process is shown in Figure 8-1. This can allow the robot to control the position of the tool endpoint and predict its visual location. These basic skills can enable rudimentary use of the tool and assist further learning by helping the robot to actively test and observe the endpoint. We show successful results for this estimation method using a variety of traditional tools shown in Figure 8-3, including a pen, a hammer, and pliers, as well as more general tools such as a bottle and the robot's own finger.

To find the tip of a tool held in the hand, the robot rotates the tool while detecting the most rapidly moving point between pairs of consecutive images. It then estimates the 3D point in the hand's coordinate system that best explains these noisy detections. Given this

protocol, motion serves as a powerful cue for detecting the endpoint of a wide assortment of human tools. The method makes few assumptions about the size and shape of the tool, its position in the robot's hand, or the environment in which the tool is being manipulated. Importantly, we use the robot's kinematic model to transform the perceptual detections into the hand's coordinate frame, allowing for the registration of many detections from multiple views. This makes the algorithm robust to noise from sources such as ego-motion and human interaction.

8.1.1 Related Work in Robot Tool Use

Research involving robot tool use often assumes a prior model of the tool or construction of a model using complex perceptual processing. A recent review of robot tool use finds few examples of robots using everyday, human tools [123]. NASA has explored the use of human tools with the Robonaut platform. Robonaut used detailed tool templates to successfully guide a standard power drill to fasten a series of lugnuts [61]. Approaches that rely on the registration of detailed models are not likely to efficiently scale to the wide variety of human tools. Williamson [143] demonstrated robot tool use in rhythmic activities such as drumming, sawing, and hammering by exploiting the natural dynamics of the tool and arm. This work required careful setup and tools that were rigidly fixed to the hand.

A long history of work in AI and computer vision has focused on learning tool function [146]. For example, Duric [38] looked at associating a tool's function with its prototypical motion. Robots that can actively learn about tool use have been the subject of more recent work. Bogoni [13] investigated relating the physical properties of the tool to the perceptual outcomes of its use when tested by a robot. Stoytchev [126] has explored learning a tool's function through its interactions with objects. This body of work typically assumes that a clean segmentation of the tool can be extracted from the image or that the tool features are known in advance.

In our method, we use our knowledge of how the robot's hand rotates while holding the tool to make 3D estimations about the location of the tool tip. This relates to methods for 3D scanning in which objects are placed on a rotating platform in front of a single camera [41]. These methods, however, typically rely on a well modeled background to cleanly segment the object, simple platform motion, and occlusion free views of the object. More generally, our estimation technique relates to the well-studied area of 3D estimation from

multiple views [56].

-XXXrelated to Danica Kragic and Christensen. Multiple cue integration for visual control.

8.1.2 Tool Tip Detection

We wish to detect the end point of a grasped tool in a general way. The detection process looks for points that are moving rapidly while the hand is moving. This ignores points that are not controlled by the hand and highlights points under the hand's control that are far from the hand's center of rotation. Typically tool tips are the most distal component of the tool relative to the hand's center of rotation, and consequently have higher velocity. The hand is also held close to the camera, so projection tends to increase the speed of the tool tip in the image relative to background motion. The wrist is rotated and we detect the tip as the fastest moving point in the image using the method of Kemp described in Section 5.1.

8.1.3 Probabilistic Estimation of the 3D Tool Tip Position

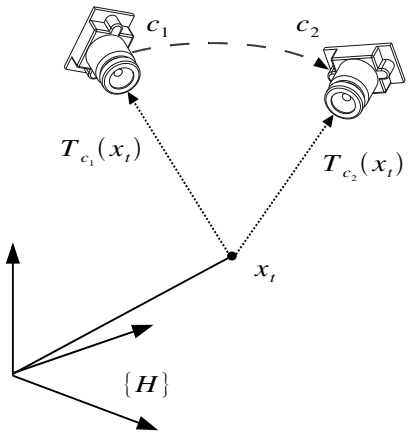


Figure 8-2: The geometry of the tool tip 3D estimation problem. With respect to the hand's coordinate system, $\{H\}$, the camera moves around the hand. In an ideal situation, only two distinct 2D detections would be necessary to obtain the 3D estimate. Given two observations with kinematic configurations c_1 and c_2, the tool tip, x_t, appears in the image at $T_{c_1}(x_t)$ and $T_{c_2}(x_t)$.

As described, visual motion is used to detect motion feature points that are likely to

correspond with the tip of the tool in the robot's hand. After detecting these points in a series of images with distinct views, we use the robot's kinematic model to combine these 2D points into a single 3D estimate of the tool tip's position in the hand's coordinate system. With respect to the hand's coordinate system, $\{H\}$, the camera moves around the hand while the hand and tool tip remain stationary. This is equivalent to a multiple view 3D estimation problem where we wish to estimate the constant 3D position of the tool tip, $\mathbf{x}_t^h = \mathbf{x}_t$, with respect to $\{H\}$. In an ideal situation, only two distinct 2D detections would be necessary to obtain the 3D estimate, as illustrated in Figure 8-2. However, we have several sources of error, including noise in the detection process and an imperfect kinematic model.

A variety of approaches would be appropriate for this estimation, since only three parameters need to be estimated and we have plenty of data from a moderately noisy source. In this paper, we estimate $\mathbf{x}_t$ by performing maximum likelihood estimation with respect to a generative probabilistic model.

We first model the conditional probability distribution, $p(\mathbf{d}_i|\mathbf{x}_t, c_i)$, which gives the probability of a detection at a location in the image, $\mathbf{d}_i$, given the true position of the tool tip, $\mathbf{x}_t$, and the robot's configuration during the detection, c_i. We model the detection error that is dependent on $\mathbf{x}_t$ with a 2D circular Gaussian, $\mathcal{N}_t$, centered on the true projected location of the tool tip in the image, $T_{c_i}(\mathbf{x}_t)$, with variance σ_t. T_c is the transformation that projects $\mathbf{x}_t$ onto the image plane given the configuration of the robot, c_i. T_{c_i} is defined by the robot's kinematic model and the pin hole camera model for the robot's calibrated camera. This 2D Gaussian error model on the image plane can coarsely represent a number of error sources, including the selection of motion edges around the ideal location, and inaccuracies in the kinematic model. We mix this Gaussian with another 2D Gaussian, $\mathcal{N}_f$, centered on the image with mean 0 and a large variance σ_f. This Gaussian accounts for false detections across the image that are independent of the location of the tool tip. In summary,

$$p(\mathbf{d}_i|\mathbf{x}_t, c_i) = (1-m)\mathcal{N}_t(T_{c_i}(\mathbf{x}_t), \sigma_t^2 I)(\mathbf{d}_i) + m\mathcal{N}_f(0, \sigma_f^2 I)(\mathbf{d}_i), \tag{8.1}$$

where m is the mixing parameter for these two Gaussians.

We model a series of detections $\mathbf{d}_1 \ldots \mathbf{d}_n$ with corresponding configurations of the robot,

$c_1 \ldots c_n$, as being independently drawn from this distribution, so that

$$p(\mathbf{d}_1 \ldots \mathbf{d}_n | \mathbf{x}_t, c_1 \ldots c_n) = \prod_i p(\mathbf{d}_i | \mathbf{x}_t, c_i). \tag{8.2}$$

Using Bayes rule we have

$$p(\mathbf{x}_t | \mathbf{d}_1 \ldots \mathbf{d}_n, c_1 \ldots c_n) = \frac{p(\mathbf{d}_1 \ldots \mathbf{d}_n | \mathbf{x}_t, c_1 \ldots c_n) p(\mathbf{x}_t, c_1 \ldots c_n)}{p(\mathbf{d}_1 \ldots \mathbf{d}_n, c_1 \ldots c_n)}. \tag{8.3}$$

Since we are only looking for relative maxima, we can maximize

$$p(d_1 \ldots d_n | \mathbf{x}_t, c_1 \ldots c_n) p(\mathbf{x}_t, c_1 \ldots c_n). \tag{8.4}$$

We also assume that the tool tip position in the hand's coordinate system is independent of the configurations of the system at which the images were captured, so that $p(\mathbf{x}_t, c_1 \ldots c_n) = p(X_h) p(c_1 \ldots c_n)$. Since $c_1 \ldots c_n$ are known and constant for the data set, we can drop their distribution from the maximization to end up with

$$\begin{aligned} \widehat{\mathbf{x}_t} &= \text{Argmax}_{\mathbf{x}_t} \left(p(\mathbf{d}_1 \ldots \mathbf{d}_n | \mathbf{x}_t, c_1 \ldots c_n) p(\mathbf{x}_t) \right) \\ &= \text{Argmax}_{\mathbf{x}_t} \left(\log \left(p(\mathbf{x}_t) \right) + \textstyle\sum_i \log \left(p(\mathbf{d}_i | \mathbf{x}_t, c_i) \right) \right). \end{aligned} \tag{8.5}$$

We define the prior, $p(\mathbf{x}_t)$, to be uniform everywhere except at positions inside the robot's body or farther than 1 meter from the center of the hand. We assign these unlikely positions approximately zero probability. A variety of methods could be used to find our estimate $\widehat{\mathbf{x}_t}$ given expression 8.5, including gradient ascent and brute force sampling. We use the Nelder-Mead Simplex algorithm implemented in the open source SciPy scientific library to optimize this cost function [70]. More efficient optimization algorithms are applicable, but this algorithm is easy to use since it only requires function evaluations. Even though each evaluation of the cost function requires $O(n)$ computation, where n is the number of detections, we found it to converge quickly given our small set of moderately noisy observations. There are many sources of error that we ignore in our model, including uncertainty about the hand's rotation (which will have a larger impact on long objects), the projection dependent aspects of the kinematic uncertainty, and uncertainty in the temporal alignment of the kinematic configuration measurements and the motion-based detections.

Figure 8-3: The approach was evaluated on a hot-glue gun, screwdriver, bottle, electrical plug, paint brush, robot finger, pen, pliers, hammer, and scissors.

8.1.4 Tool Tip Estimation Results

Experiments were conducted on a variety of tools with differing lengths and endpoints (Figure 8-3). For each experiment, the 11 DOF kinematic chain from the camera to the robot wrist was servoed to maintain a fixed pose that ensured tool visibility in the wide-angle camera. The tool was placed in the robot's hand and the 2 DOF (pitch,roll) of the wrist were ramped smoothly to random positions in the range of ± 60 degrees for a short duration. The robot's joint angles and camera images were sampled at $30hz$. Approximately 500 samples (15 seconds of motion) were captured for each tool and randomly distributed into a training set of 400 samples and a test set of 100 samples. We then hand labeled the tool tip location for each frame of the test set.

Tool Tip Detection

Visual detection of the tool tip was computed using the motion model from Section 5.1. In these experiments, the localization was computed offline for each pair of sequential images, though real-time rates are achievable. A naturally lit, cluttered background was used (Figure 8-1A) to ensure a non-trivial environment for perception and background motion was allowed. The detection method is noisy, but as shown in Figure 8-7, the detections tended to match the hand-labeled tool tip locations. In the experiments we present, the camera and environment were nearly stationary and the affine model of background motion was estimated as close to identity. Without modification the algorithm can be used in situations

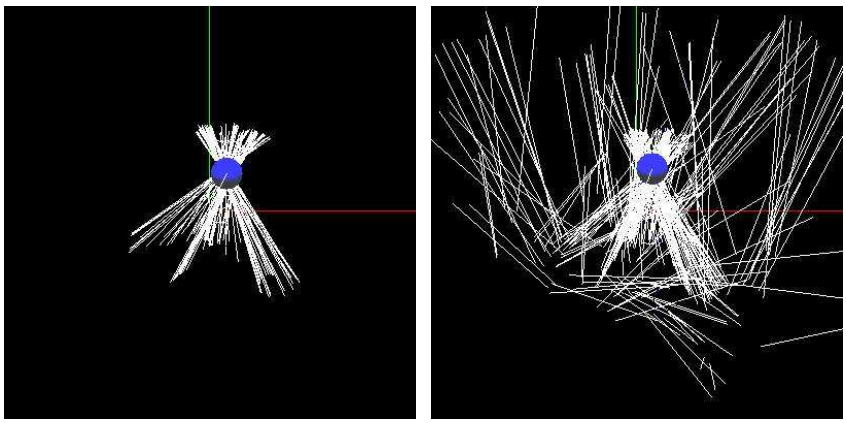


Figure 8-4: As the robot rotates a grasped object in front of its camera, its attention system generates monocular detections of the distal end of the object. Each detection is transformed into a ray within the hand's coordinate system (white). Many such rays intersect at the estimated tip location (blue). The left image visualizes these rays without background motion. The right image is noisier due to background motion, but the tip location is still robustly estimated.

with a slowly moving camera and other causes of global affine motion.

Tool Position Estimation

The position estimation accuracy was evaluated by first estimating the 3D tool position in the hand on the training data set as described in Section 8.1.3. We used the parameter values $\sigma_t = 5.0$, $\sigma_f = 150.0$, and $m = 0.5$. The 3D position was projected onto the image plane for each sample in the test set. The predicted tool tip location was measured against the hand labelled location to compute the mean pixel error. A baseline comparison can be made by performing the estimation process on the hand labeled data set. The resulting error is indicative of inaccuracies in the kinematic model and the camera model. The algorithm performs favorably with respect to this baseline error. Figure 8-5 illustrates the mean prediction error, in pixels, across the set of tools. Figure 8-6 illustrates the typical tip prediction for each tool.

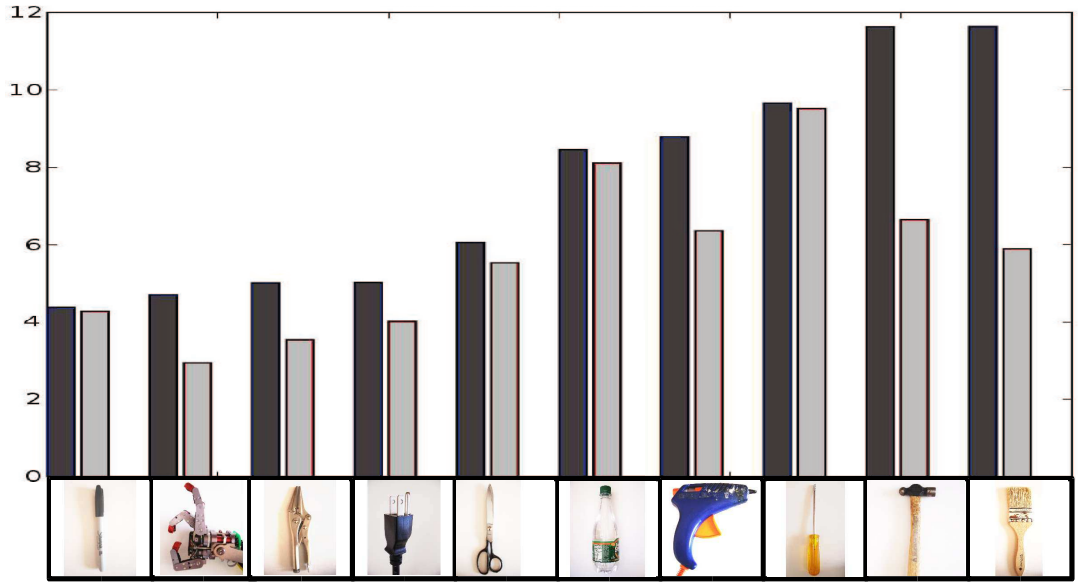


Figure 8-5: The mean prediction error, in pixels, for each tool. The 3D tool tip position in the hand is estimated using two data sets: 2D motion-based tool tip detection and hand-labelled tool tips. The 3D positions for both estimates are then projected onto the image plane for each sample in the test set and compared to the hand labelled location. The left (dark) bar indicates the detector error and the right (light) bar indicates the hand labelled, baseline error. The baseline errors are an indication of inaccuracies in the kinematic model and the camera model.

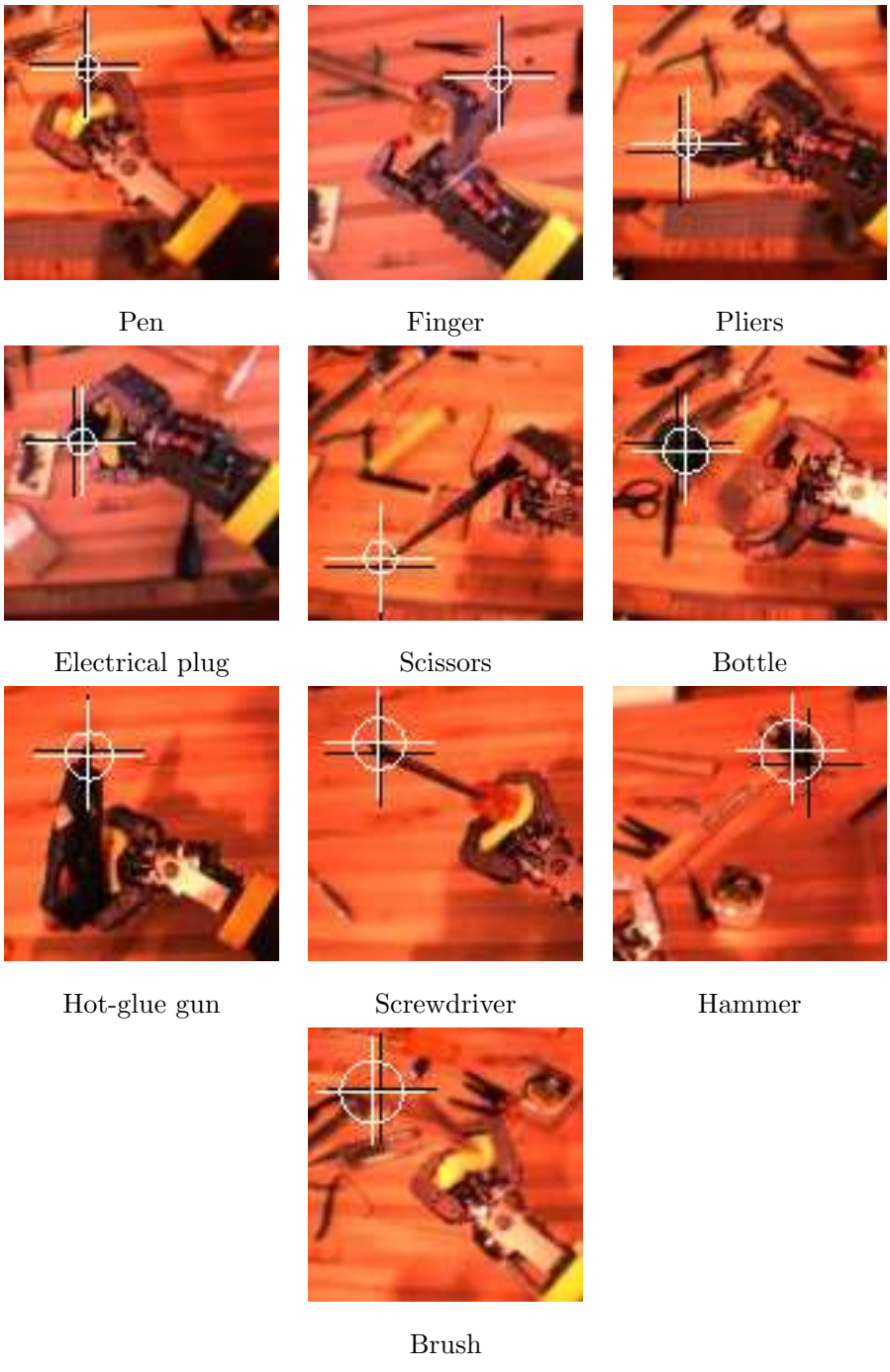


Figure 8-6: An example of the tip prediction for each tool. The white cross is centered at the prediction point and measures 40 pixels across for scale. The radius of the white circle indicates the tool's mean pixel error. The black cross indicates the hand labeled tool tip.

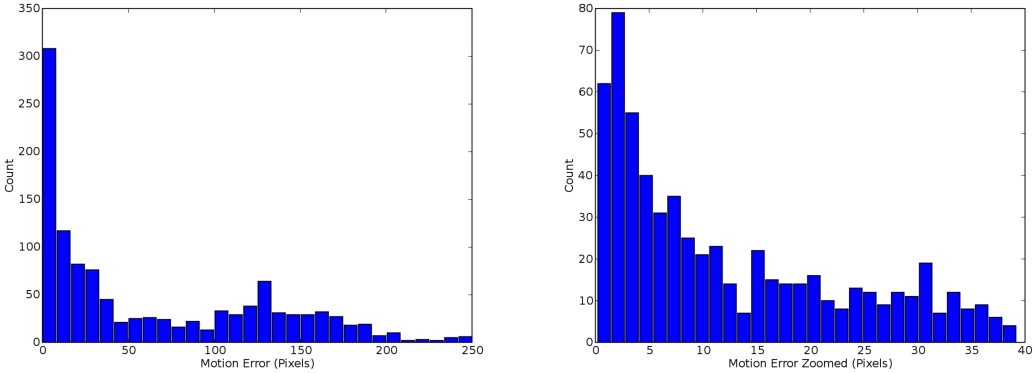


Figure 8-7: Error histogram, in pixels, for raw motion-based detection of the tool tip with respect to the hand-labeled tool tip for all tools. [Right] Detailed view of the left graph.

Discussion of the Results

As Figure 8-6 illustrates, the estimation process performs well as measured by the image prediction error. The wide angle camera from which the images were captured allows a larger variety of tool sizes to be explored, but the resolution of the tip was often low, on the order of 10 pixels. Errors can originate from poor calibration in the kinematic and camera model, as the baseline errors in Figure 8-5 demonstrate. We trained each estimator on a data set of 400 samples which is conservatively high given the effectiveness of the motion-based detector and the ideal requirement of only two distinct views. It is important that the wrist sample a large space of poses. In the extreme case of hand rotation occurring only in the image plane, the depth of the tool position would be indeterminate.

8.1.5 Discussion

In this section we presented a straight-forward approach to detecting the end point of a tool in a robot hand and estimating its 3D position. The strength of our approach is that it assumes little prior knowledge about the tool or its position in the hand and avoids complex perceptual processing. Rather than segmenting the tool, estimating the 3D shape of the tool, or otherwise representing the details of the tool prior to detecting the tip, this method jumps directly to detecting the tip of the tool. The success of the method relies on two main observations. First, the natural utility of many human tools depends on the tool's

endpoint. Second, for many of these tools the endpoint can be detected by its rapid motion in the image when the robot moves its hand while holding the tool. For the results we present, the robot's hand is roughly human in size and shape and thus well-matched to human tools. This detection method might not perform as well with robot end-effectors that differ significantly from a human hand (for example they might be large with respect to the tool).

We estimate the 3D tool tip position in batch-mode by optimizing the cost function of Equation 8.5. A recursive filter, such as an extended Kalman filter, could provide an adaptive, online alternative to the estimation technique. This could be used to adjust for possible slip in the robot's grasp of the tool during use. As we will see in the Section 8.3, other perceptual cues can be beneficially integrated into this method. Motion does, however, have some especially beneficial properties for this type of detection. First, motion helps us to find elements of the world that are controlled by the robot's hand. Stereo analysis of a static scene could be used to select elements of the scene that are close to the hand, but without motion, stereo could not detect which points are under the hand's control. Second, by moving the hand and tool we are able to observe them from several distinct views, which reduces sensitivity to the particular position of the hand and increases overall robustness.

8.2 Control of the Tip

8.2.1 *TipPose*

The *TipPose* module places the tip of a grasped tool at a desired position and location in the world frame. The 3D estimate of the tip's location within the hand's coordinate frame, $\widehat{\mathbf{x}}_t$, can be used to effectively extend the robot's kinematic model by a link. This provides many options for control. In this section we describe a virtual spring method for control of the position and orientation of this task-relevant feature. Virtual spring control of the tooltip uses the well known Jacobian transpose method [35] for relating manipulator forces to torques. It is a straightforward technique strongly related to virtual model control [68] and operation space control [77]. Virtual forces at the end-effector are simulated that bring the tool tip to a desired pose. This linear method typically assumes that the joint angles start close to their final state.

The Jacobian, $\mathbf{J}$, is known from the kinematic model. It relates a $6x1$ force-moment

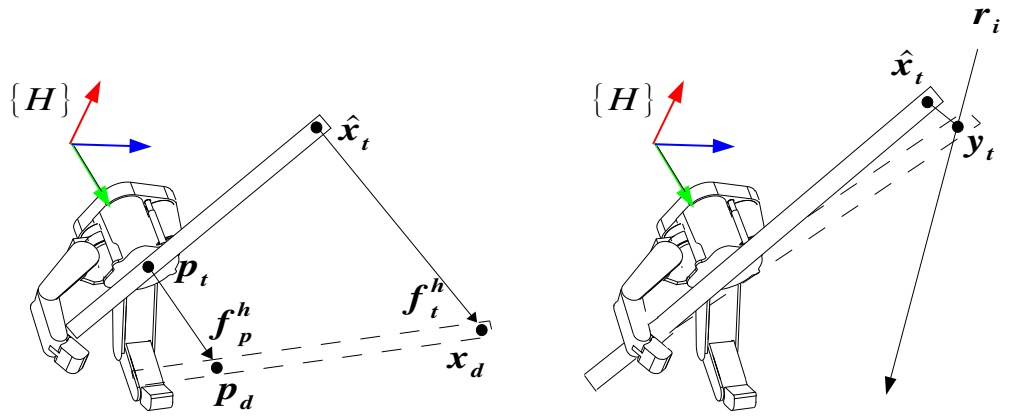


Figure 8-8: **Left**: Virtual springs control the position of the palm $\mathbf{p}_t$ and the tooltip $\widehat{\mathbf{x}}_t$ in the hand's coordinate frame $\{H\}$. Forces $\mathbf{f^h}_t \propto (\mathbf{x}_d - \widehat{\mathbf{x}}_t)$ and $\mathbf{f^h}_p \propto (\mathbf{p}_d - \mathbf{p}_t)$ generate a virtual wrench at the end-effector which achieves the desired tip location $\mathbf{x}_d$ and the desired palm location $\mathbf{p_d}$. **Right**: Visual tracking of the tooltip defines a ray, r_i, in $\{H\}$. The closest point to $\widehat{\mathbf{x}}_t$ on the ray is $\mathbf{y}_t$. This location is used to visually servo the tip to a target.

wrench at the hand to joint torques as $\tau = \mathbf{J^T f}^w$. Instead of controlling the arm's joint torque directly, we control the joint angle, and our controller takes the form of $\Delta\theta = \sigma \mathbf{J}^T \mathbf{f}^w$ for controller gain σ.

Using this controller, the position and orientation of the tip of a grasped object can be controlled by specifying the virtual wrench $\mathbf{f}^w$ at the end-effector. As shown in Figure 8-8, this wrench is created through two virtual springs in the hand's coordinate frame $\{H\}$. One spring controls the position of the tip by moving $\widehat{\mathbf{x}}_t$ to the target location, $\mathbf{x}_d$. The other spring controls the orientation of the tip by moving the robot's palm, $\mathbf{p}_t$, to the target location $\mathbf{p}_d$.

To compute the net wrench $\mathbf{f}^w$, we first introduce the force-moment transform [35],

$$M(\mathbf{R}, \mathbf{t}) = \begin{bmatrix} \mathbf{R} & 0 \\ P(\mathbf{t})\mathbf{R} & \mathbf{R} \end{bmatrix}, \tag{8.6}$$

where $P(\mathbf{t})$ is the cross product operator

$$P(\mathbf{t}) = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}. \tag{8.7}$$

Given the coordinate transform ${}^a\mathbf{T}^b = [{}^a\mathbf{R}^b | \mathbf{t}^a_{borg}]$, the force-moment transform relates a wrench in $\{B\}$ to a wrench in $\{A\}$ as $M({}^a\mathbf{R}^b, \mathbf{t}^a_{borg})$. We can now relate a virtual wrench at $\widehat{\mathbf{x}}_t$ to $\{H\}$ using $M(\mathbf{I}, \widehat{\mathbf{x}}_t)$. Likewise, a virtual wrench at the palm relates to $\{H\}$ through $M(\mathbf{I}, \mathbf{p}_t)$. Therefore, each virtual spring generates a virtual wrench at the wrist as

$$\mathbf{f}^h_t = \mathbf{K}_t M(\mathbf{I}, \widehat{\mathbf{x}}_t) \begin{bmatrix} (\mathbf{x}_d - \widehat{\mathbf{x}}_t) & 0 & 0 & 0 \end{bmatrix}^T \tag{8.8}$$

$$\mathbf{f}^h_p = \mathbf{K}_p M(\mathbf{I}, \mathbf{p_t}) \begin{bmatrix} (\mathbf{p}_d - \mathbf{p}_t) & 0 & 0 & 0 \end{bmatrix}^T, \tag{8.9}$$

The spring stiffness $\mathbf{K}_t$ and $\mathbf{K}_p$ is adjusted in a diagonal gain matrix. We can align these forces with $M({}^w\mathbf{R}^h, 0)$, giving the desired virtual wrench acting on the end-effector:

$$\mathbf{f}^w = M({}^w\mathbf{R}^h, 0) \left(\mathbf{f}^h_t + \mathbf{f}^h_p \right). \tag{8.10}$$

This controller assumes a manipulator with a spherical 3 DOF wrist, allowing arbitrary rotation and translation of the hand frame. It also assumes that the manipulator is not near a singularity. Domo's wrist has only 2 DOF and consequently we must assume that the correct orientation is locally achievable with the restricted kinematics. If we initialize this controller with a task-dependent manipulator pose, then in most cases it is possible to avoid singularities and issues with the restricted kinematics. Related techniques such as Damped Least Squares could be used instead of the Jacobian transpose in order to limit these restrictions [26].

8.2.2 *TipServo*

Visual servoing can allow a manipulator to be controlled using a coarsely calibrated hand-eye transform. There are numerous techniques for accomplishing this. An excellent survey of visual servoing for manipulation is provided by Kragic [81]. In the absence of visual servoing, a robot will typically map a visual scene to a 3D object model in the camera frame, transform this model to the world frame, and then use a cartesian space controller to

adjust its manipulator relative to the object. This approach requires a precisely calibrated models which is not always easy, or possible to produce. In addition, a 3D model of the object isn't always available, particularly in human environments.

TipServo incorporates visual feedback into the *TipPose* module in order to compensate for Domo's coarsely calibrated hand-eye transform. This coarse calibration can cause errors in the estimation of the tool tip $\widehat{\mathbf{x}}_t$. Estimate $\widehat{\mathbf{x}}_t$ minimizes the pixel error between feature detections and the projection of $\widehat{\mathbf{x}}_t$ into the image. Consequently, even if $\widehat{\mathbf{x}}_t$ fits the detection data well, it can be inaccurate with respect to a coarsely calibrate 3D kinematic frame. As the arm moves away from the pose where $\widehat{\mathbf{x}}_t$ is learned, these errors become evident. An open-loop controller like *ToolPose* is succeptible to these errors, especially in fine-resolution tasks such as the insertion of two similar sized objects.

TipServo visually detects the tip and adjusts $\widehat{\mathbf{x}}_t$ online. The algorithm is summarized as follows:

1. Visually detect the tip whenever the wrist rotates as detection $\mathbf{d}_i$.
2. Compute the probability that the detection corresponds to the tip as $p(\mathbf{d}_i|\widehat{\mathbf{x}}_t, c_i)$ using Equation 8.1.
3. Whenever $p(\mathbf{d}_i|\widehat{\mathbf{x}}_t, c_i) > \epsilon$, for some threshold ϵ, (re)initialize a visual feature tracker using $\mathbf{d}_i$.
4. Use the 2D tracked feature to create an online approximation of the 3D tip location $\mathbf{y}_t$.
5. Substitute $\mathbf{y}_t$ for $\widehat{\mathbf{x}}_t$ in the feedforward *TipPose* controller.

It should be pointed out that the manipulator is servoed in 3D using 2D information. This type of controller is classifed by Kragic as a position-based-visual-servo-system [81]. As shown in Figure 8-8, we define $\mathbf{y}_t$ as the point on the ray $\mathbf{r}_i$ that is closest to $\widehat{\mathbf{x}}_t$, where $\mathbf{r}_i$ is the ray in the hand frame that passes through the tracked feature. This is equivalent to choosing the maximum likelihood location when the estimation error is distributed around $\widehat{\mathbf{x}}_t$ according to a 3D Gaussian model. Of course, the actual distribution is a complex function of the kinematic state, but this simplification suffices for our purposes. We don't control the tracked image feature $\mathbf{r}_i$ directly for two reasons. First, we would like reuse the *TipPose*

controller. Second, a low update rate by the tracker can cause controller instability. $\mathbf{y}_t$ is a stable point in $\{H\}$ so long as the error changes slowly across the manipulator workspace.

8.2.3 Results

As shown in Figure 8-9, we tested the virtual spring controller with and without visual feedback. The robot rigidly grasped a long spoon with a color fiducial marking the tip. The tip location was then estimated as $\widehat{\mathbf{x}}_t$. In the first experiment, the feedforward virtual spring controller traced the tip along an 100 pixel square in the image at a fixed depth from the camera of 350mm . The desired orientation of the spoon was aligned with gravity. The controller error was measured between the location of the fiducial in the image and the desired tip location on the square. In the second experiment, visual feedback was introduced into the controller. The tip $\widehat{\mathbf{x}}_t$ was first servoed to the start point on the square. Spoon motion was generated at the wrist for a period of **5** seconds in order to create an initial estimate of the tip estimation error $\mathbf{e}^h$. The tip $\mathbf{y}_t = \widehat{\mathbf{x}}_t + \mathbf{e}^h$ was then traced around the square using the virtual spring controller. During this time, the spoon's motion was used to update $\mathbf{e}^h$. As the results show, the visual feedback significantly reduced the feedforward controller error. Inspection of the feedforward controller trajectory shows a horizontal offset in the estimation of $\widehat{\mathbf{x}}_t$. The visual feedback compensates for this offset.

8.3 Moving from Tooltips to Everyday Objects

Previously we described a method for the autonomous detection of the tip of a grasped tool and the estimation of the tip's position in the robot's hand. The approach was demonstrated on a range of tools, including a screwdriver, pen, and a hammer. For these objects, we assumed that a tool is characteristically defined by a single, distal point. However, this characterization does not fit objects such as cups, bowls, jars, brushes, pots, and bottles. These are not tools in a traditional sense, yet they still have a tip or endpoint that is of primary importance during control.

In this section, we extend our definition of a tooltip. We define the tip of a tool as occupying a circular area of some radius. In particular, we use the *InterestRegions* module to detect rapidly moving edges that are approximately tangent to a circle of some radius. This detector performs well on objects that do not have a sharp point, allowing us to expand

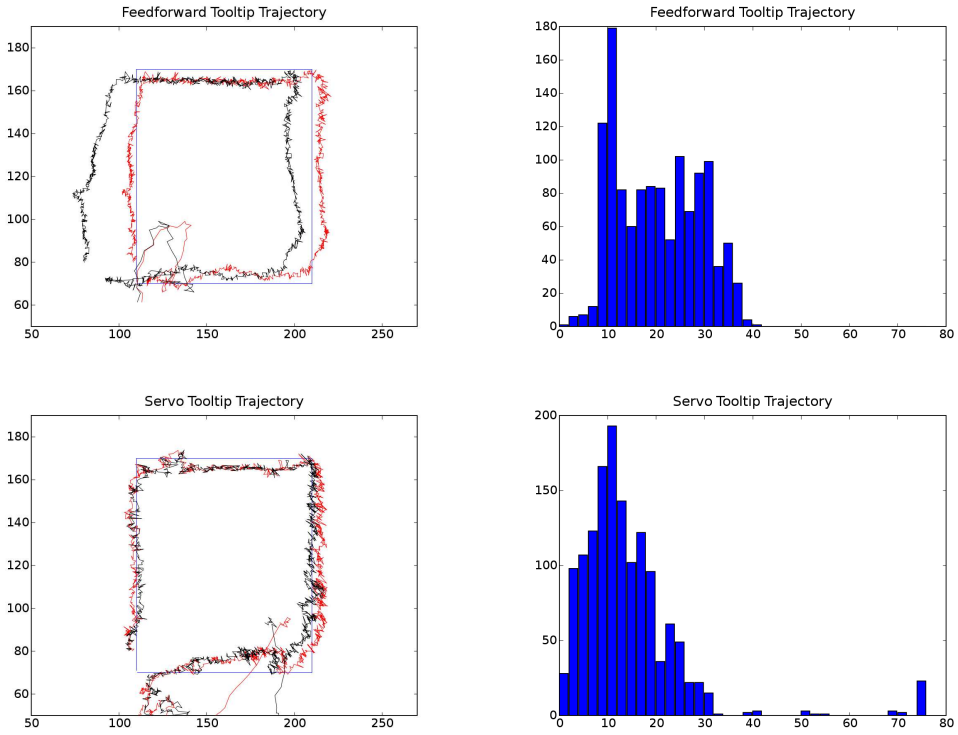


Figure 8-9: Results from execution of the *TipPose* (top) and *TipServo* (bottom) modules. After estimating the tip $\widehat{\mathbf{x}}_t$, each module traced the tip, projected into the image (red, left), around an 100 pixel square (blue, left). The actual tip location was sensed in the image with a color fiducial on the tip(black, left). The histograms (right) show the pixel error between the desired square and the fiducial tip. **Top**: Using *ToolPose*, hand-eye calibration errors cause the fiducial tip location to differ from $\widehat{\mathbf{x}}_t$ in the image. **Bottom**: *TipServo* uses visual feedback to reduce the fiducial error.

our notion of the tip of an object to include such items as a bottle with a wide mouth, a cup, and a brush. This feature detector outperforms our previous method on these three objects. It also estimates the scale of the tip which can be used to build a visual model.

We use the same protocol as our tooltip detection method. An object is grasped and rotated at the wrist. Given a pair of sequential images, *InterestRegions* computes the interest point detection map, m_s, from Equation **??**. For each scale-space s, we detect the points within m_s that have a strong response above a threshold. After n detections $\mathbf{d}_1 \dots \mathbf{d}_n$, the detections are passed to the tooltip estimation algorithm and $\widehat{\mathbf{x}}_t$ is computed as before.

The size of the object's tip can be estimated using the scale of each detection provided by *InterestRegions*. First, for detection d_i we use the kinematic configuration c_i to compute the depth of $\widehat{\mathbf{x}}_t$ in the camera frame $\{C\}$ as z_i. The tip size for d_i is then $\frac{2r_{si}z_i}{f}$ for the scale radius r_{si} and camera focal length f. Next, we select all probable detections such that, given d_i and threshold ϵ, $p(\mathbf{d}_i|\widehat{\mathbf{x}}_t, c_i) > \epsilon$. Finally, the estimated size of the object's tip is taken as the average tip size for all probable detections.

8.3.1 Results

We validated our method on a bottle, a cup, and a brush, as pictured in Figure 8-5. The items were chosen for their varying tip size and length. When the tool is placed in the robot's hand, it automatically generates a short sequence of tool motion of about 200 samples over 5 seconds.

For each tool we compare the multi-scale detector to the original edge-motion detector. Figure 8-5 shows the mean prediction error, as measured by the tool tip projection into the image, for the two detectors. The multi-scale detector significantly improves the predicted location for these three objects that have large, broad tips. The detector also enables online modeling of the tip. Figure 8-10 shows the ability of the detector to appropriately extract the size of the tool tip. Figure 8-11 illustrates the construction of a pose normalized visual model of the tip. This model facilitates the use of visual features that describe the appearance of the tip over its estimated spatial extent. For example, an HSV histogram of the tip apperance could be learned online to augment the detectors performance.

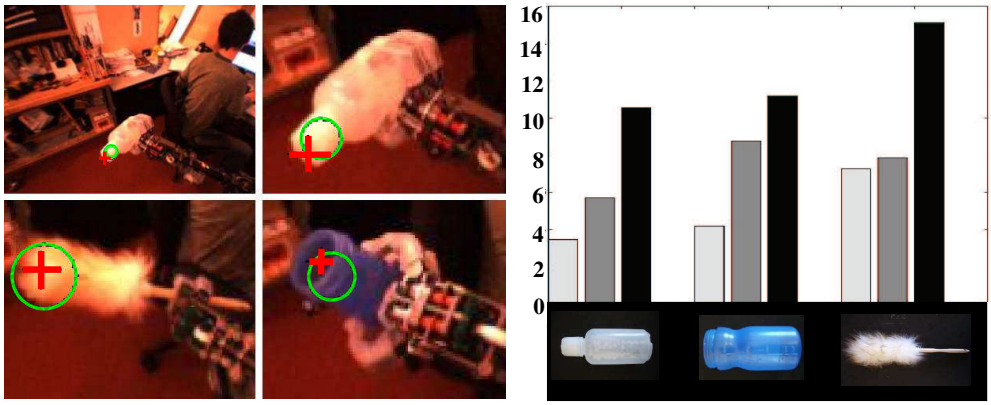


Figure 8-10: Detection and estimation of the generalized tooltip for three test objects. **Left**: The upper left image gives an example of the images used during estimation. The movement of the person in the background serves as a source of noise. The red cross marks the hand annotated tip location and has a width equivalent to twice the mean pixel error for prediction over the test set. The green circle is at the tip prediction with a size equal to the estimated tip size. **Right**: The mean prediction error, in pixels, for each of the three tools. The 3D tool pose is estimated in three ways: the hand labelled tips (left bar), detections from *InterestRegions* (middle bar), and the maximum motion point (right bar).

Figure 8-11: These average tip images give an example of acquiring a model of the tip's appearance. For the three test objects, square image patches of the tips were collected using the tip detector, tip predictor, and smoothing of the estimated state. They were then normalized in scale and orientation and averaged together.

8.4 Learning the Task Specific Prior

For many tasks and objects, a task relevant feature will appear in a canonical region relative to the hand. When we write with a pen or use a hammer, we have a strong prior expectation of where the functional end of these objects will appear. A robot could learn these priors from experience, allowing it to more quickly localize the feature and discount features far from the prior. Even though the exact location of the task relevant feature will vary depending on the object and grasp, we assume that the feature location can be usefully estimated with a probability distribution. Such priors are task specific. A water bottle that is being poured may have a different prior than a water bottle that is being placed on a shelf.

Learning the task specfic prior also allows the robot to predict where the feature might appear in the image. This relates to the work of Torralba [133], who describes a framework that uses contextural priors to improve visual search and modulate the saliency of image regions. Prior knowledge of where the feature may be can aid the robot's controllers. In order to localize a tool tip using a uniform prior, Domo must hold the object far from the camera and randomly sample from the full range of wrist postures. Given a non-uniform prior, Domo can now bring the object closer to the camera and only sample from wrist postures that provide unobstructed views of the maximum likelihood tip location.

In Section 8.1.3 we defined the prior on the expected tool tip location, $p(\mathbf{x}_t)$, to be uniform everywhere except at positions inside the robot's body or farther than 1 meter from the center of the hand. In this section we consider how Domo can learn a non-uniform prior $p(\mathbf{x}_t|q)$ conditioned on the specific task category q . This estimation process is illustrated in Figure 8-12 .

8.4.1 Density Estimation

We estimate the task specific prior, $p(\mathbf{x}_t|q)$, using a probability distribution centered around the robot's hand. An 3D histogram, or occupancy grid, is used to estimate the distrubtion. For a specific category of task relevant feature q , the robot manipulates a small set of typical objects used in that task. To detect the feature as in Section 8.1.2, each object is grasped and the robot rotates it while detecting the most rapidly moving point between pairs of consecutive image. Each visual detection corresponds to a ray in the hand's coordinate

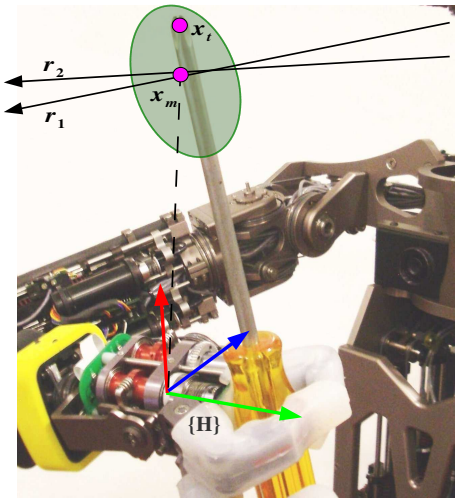


Figure 8-12: The task relevant prior, $p(\mathbf{x}_t|q)$, is a non-uniform probability distribution centered on the hand frame $\{H\}$ for task q. It is a 3D spatial distribution (green) describing the likely location of a task relevant feature, $\mathbf{x}_t$, such as the tool tip. Two feature detections, represented by rays $\mathbf{r}_1$ and $\mathbf{r}_2$, define a candidate tip location through the nearest point of intersection, $\mathbf{x_m}$. A 3D histogram of these locations describes the non-parametric density distribution.

frame that is presumably near the distal tip of the object. In the ideal case, any two rays would intersect at the object tip. In the non-ideal case, we can take the closest point between any two rays as a candidate location of the tip. The spatial distribution of these candidate locations, accumulated over the set of task related objects, is then used to approximate $p(\mathbf{x}_t|q)$.

Collecting 3D Detections

We first find the point $\mathbf{x}_m$ that minimizes the distance between two rays. A ray is defined with a start point $\mathbf{s}$ and a point on the ray $\mathbf{p}$. The set of points on a ray are defined by $\mathbf{x}_\alpha = (\mathbf{p} - \mathbf{s})\alpha + \mathbf{s}$ such that $\mathbf{x}_\alpha \cdot (\mathbf{p} - \mathbf{s}) > 0$. We can find the points on the rays that are closest to one another using the following equation in the standard linear least squares form $\mathbf{Ax} = \mathbf{b}$.

$$\begin{bmatrix} \mathbf{p}_1 - \mathbf{s}_1 & \mathbf{p}_2 - \mathbf{s}_2 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \mathbf{s}_2 - \mathbf{s}_1,$$

If the points associated with $\alpha_1$and α_2 are valid points on the rays, then $\mathbf{x}_m = \frac{1}{2}(\mathbf{x}_{\alpha_1} + \mathbf{x}_{\alpha_2})$ defines a point that minimizes the squared distance between the two rays. This gives the maximum likelihood estimate for the intersection of the two rays given our probabilistic model[1]. The distance between the rays is also defined as $\|\mathbf{x}_{\alpha_1} - \mathbf{x}_{\alpha_2}\|$.

Now, given a visual tip detection in the image, we can define a ray in the hand frame $\{H\}$ which passes through the detection pixel and the camera focal point. If the robot manipulates an object and generates n visual detections, we can construct $\frac{n(n-1)}{2}$ pairs of rays. We collect all pairs of rays that come within some distance, k_{ray}, of one another. Pairs beyond this threshold are unlikely to have visual detections corresponding to the same feature. This computation is $O(n^2)$.

Each pair of rays corresponds to a candidate tip location in $\{H\}$. For a feature category q, candidate locations are collected across all objects in the category training set. This results in the training data set L_q which describes a spatial distribution around the true tooltip. There are many sources of error that we ignore, including error sensitivity as a function of distance from the camera due to projection, uncertainty about the hand's rota-

[1]This will be the maximum likelihood location given two detections for a large class of probabilisitic models that depend on the distance between the point and the rays.

tion that will have a larger impact on long objects, and the higher likelihood of intersections at points that are close to the camera. However, this method is computationally efficient, easy to visualize, and produces good results.

3D Histogram

For each data set L_q, we model the spatial distribution of the task relevant feature using the probability distribution $p(\mathbf{x}_t|q)$. This represents the chance, given task q, of seeing the feature at a location, $\mathbf{x}_t$ in the hand's coordinate system $\{H\}$. Using this coordinate system allows for visual tip detections to be registered across multiple views of the same object, and across multiple objects that could be applied to the same task. $p(\mathbf{x}_t|q)$ is estimated using a $k_b \cdot k_b \cdot k_b$ bin 3D histogram. The histogram spatially corresponds to a cube centered on the hand with size k_d per side. Therefore, $\frac{k_d}{2}$ is the maximum possible extent of any grasped object. We define the histogram binning function as

$$g(\mathbf{a}, \mathbf{b}) = \delta(\text{round}\left(\frac{k_b}{k_d}(a_x - b_x)\right))\delta(\text{round}\left(\frac{k_b}{k_d}(a_y - b_y)\right))\delta(\text{round}\left(\frac{k_b}{k_d}(a_z - b_z)\right)), \tag{8.11}$$

where $\delta(d) = \begin{cases} 1 \text{ if } d = 0 \\ 0 \text{ otherwise} \end{cases}$. The probability distribution is then

$$p(\mathbf{x}_t|q) \approx \frac{1}{\sum_{i \in L_q} w(i)} \sum_{i \in L_q} w(i) g(i, \mathbf{x}_t). \tag{8.12}$$

The weighting function $w(i)$ assigns a zero probability to candidate locations that are too close (k_m) or to far (k_d) away from the hand, such that $w(i) = \begin{cases} 1 \text{ if } k_m < \|i\| < k_d \\ 0 \text{ otherwise} \end{cases}$. The maximum likelihood location of the task feature is then taken as the maximal histogram bin, or $\mathbf{x}_q = \text{Argmax}\,(p(\mathbf{x}_t|q))$.

8.4.2 Results

We estimated $p(\mathbf{x}_t|q)$ for four task categories using three objects for each category:

1. Placing a mug, paper cup or watter bottle on a shelf.

2. Covering a surface with a duster, large paint brush, or small hand broom.

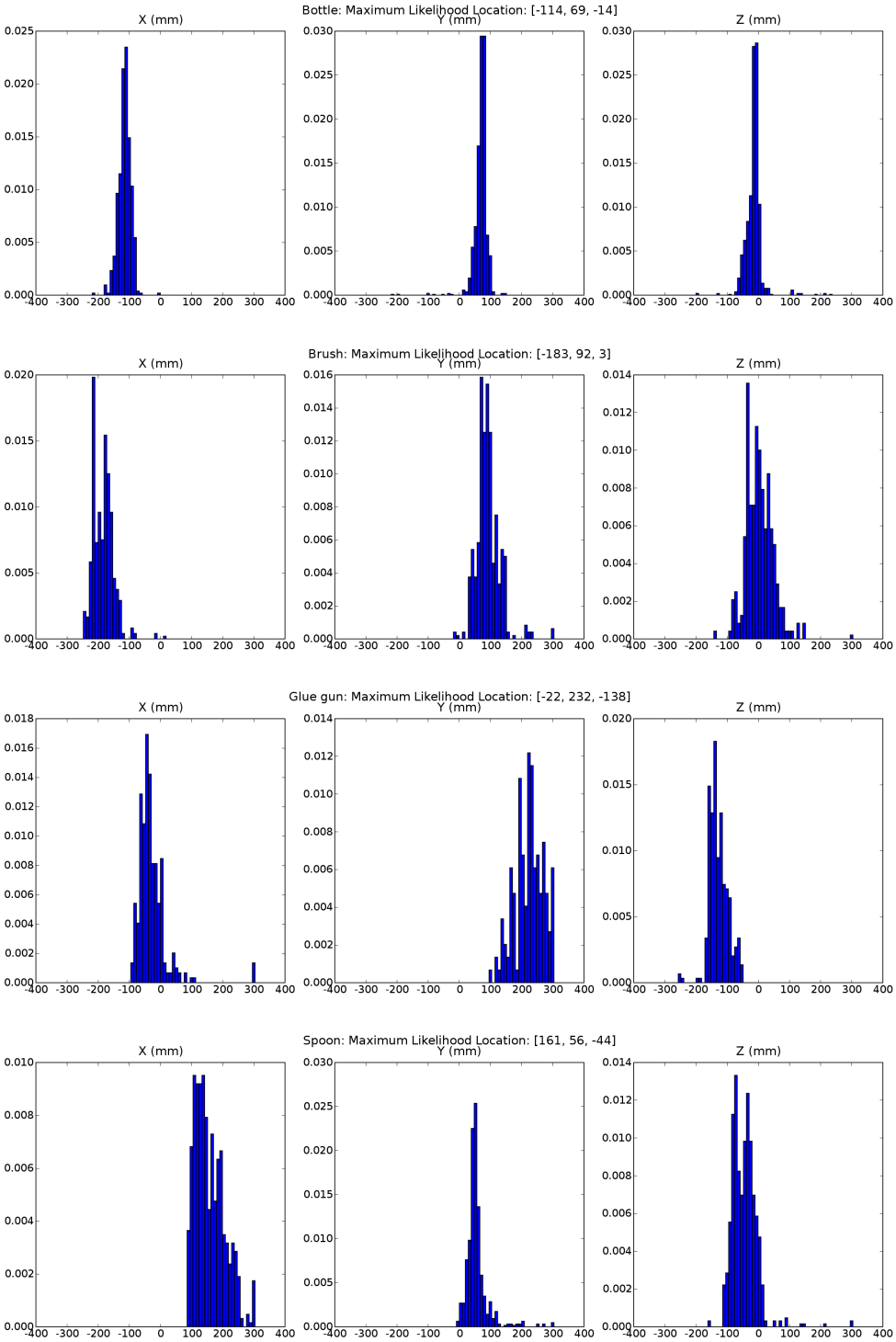


Figure 8-13: Probability distributions for the task specific priors.

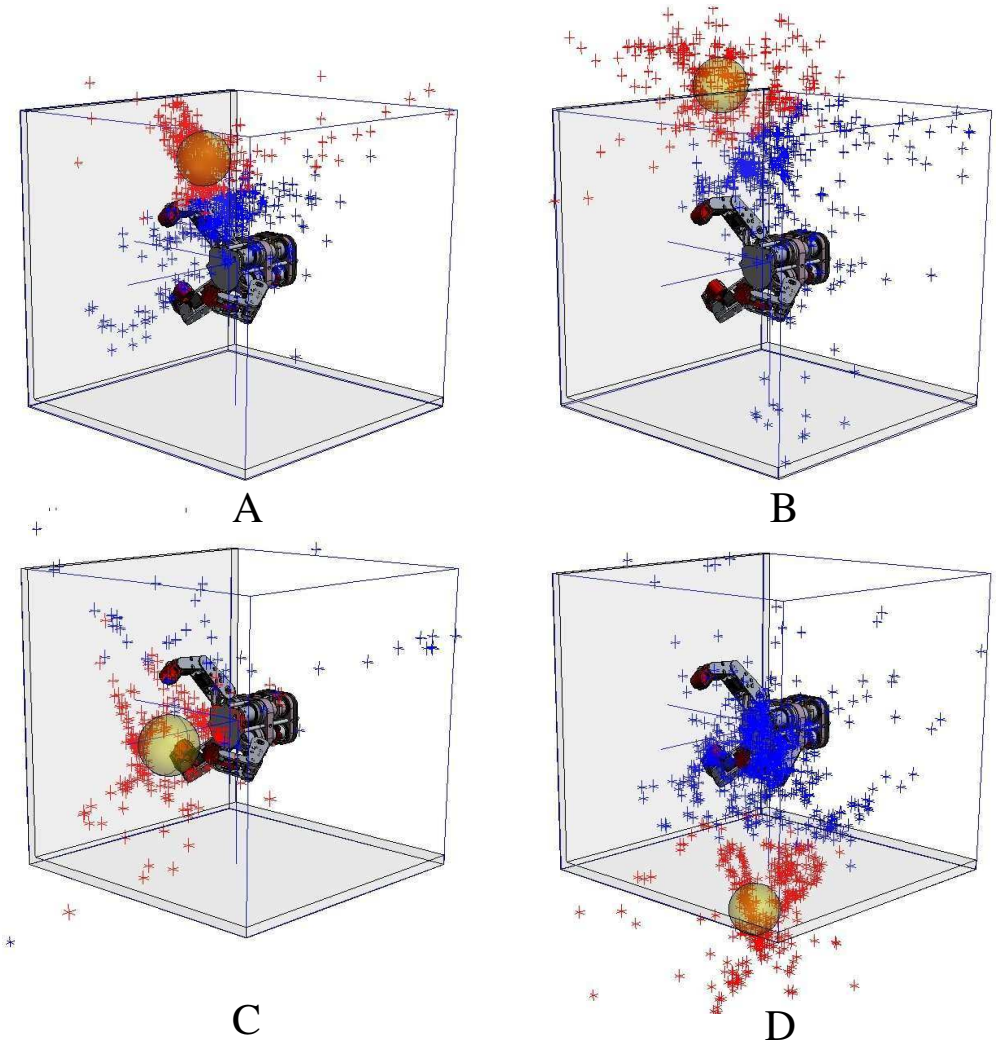


Figure 8-14: Results of learning the task specific priors for four different tasks. Each view shows a scatter plot of the 3D data L_q relative to the hand. Each data point is a candidate location of the tip of the grasped object. For each task, three different but functionally related objects were used. The data is partioned into two clusters. The distal cluster (red) is used to construct the prior density distribution for the task feature. The centroid of the cluster, depicted with a 50mm sphere, is the learned canonical location of the feature for the task. The cube is 300mm per side for scale. (A) Grasping a cup or bottle for placement on a shelf. (B) Grasping a duster or paint brush. (C) Grasping a hot glue-gun or small hand saw. (D) Grasping a stirring spoon.

3. Pointing a hot-glue gun, a heat gun, or a wooden toy gun.

4. Inserting a large stirring spoon, small stirring spoon, or condiment bottle into a container

These tasks were chosen for the unique, canonical location of the distal tip of the grasped object during task execution. For each object, the 11 DOF kinematic chain from the camera to the robot wrist was servoed to maintain a fixed pose that ensured tool visibility in the wide-angle camera. The tool was placed in the robot's hand and the 2 DOF of the wrist were ramped smoothly to random positions in the range of ± 60 degrees for a short duration. Visual detection of the fastest moving point was used to generate 50 detections of the object tip. For each of the approximately 1200 pairs of detections, $\mathbf{x}_m$ was computed and added to the dataset L_q subject to $k_{ray} = 10mm$. The histogram was constructed using $k_d = 600mm$ and $k_m = 50mm$.

We implemented an additional constraint on each valid pair of detections for practical reasons. If the two features were detected from similar wrist postures, then the two rays may be near parallel. This magnifies sources of kinematic and visual error. To reduce this effect, we found it useful to discard detection pairs when the wrist rotated less than 15 degrees between samples. It was also found useful to first cluster L_q into two clusters using K-means. The visual detection algorithm often detects features on the hand when the object tip is obstructed by the hand. Consequently, a significant component of L_q is 3D locations on the robot's hand. By first discarding the cluster that is closest to the hand, we reduce this effect.

Figure 8-14 shows the spatial distribution of these clusters around the hand for each task category. As shown, the maximum likelihood location, $\mathbf{x}_q$, corresponds to the expected object tip location for each task. The presence of noise is due to false detections typically resulting from ego-motion of the head, occlusion by the hand, or a person moving in the background. False detections tend to be randomly distributed when viewed from the moving coordinate frame $\{H\}$. In contrast, tip detections are stationary in $\{H\}$ and therefore localized around the true tip $\mathbf{x}_t$.

8.5 Working with Tips

In the previous sections we presented the components required for a robot to localize and control the tip of a unknown grasped object. This ability is integrated into the *TipUse* module so that whenever the robot is handed a new object, it can quickly find the object tip and begin controlling it for a task. In this section we describe *TipUse* and the modules required for its execution.

8.5.1 *TipPriors*

Using the task specific prior, $p(\mathbf{x}_t|q)$, from Equation 8.12, the *TipPriors* module simply computes the expected tip location $\mathbf{x}_q = \text{Argmax}\left(p(\mathbf{x}_t|q)\right)$ given the current task category q. The active high-level module with control of the manipulator will define the category q. For example, when *SurfacePlace* is activated, *TipPriors* will compute

$$\mathbf{x}_q = \text{Argmax}\left(p(\mathbf{x}_t|SurfacePlace)\right).$$

The prediction $\mathbf{x}_q$ is then used by any module that *SurfacePlace* might activate such as *TipEstimate* and *TipPose*. Although *TipPriors* limits the generality of *ToolUse* and is not required, it allows for greater efficiency when estimating $\widehat{\mathbf{x}}_t$. This is convienient for real-time tasks involving a collaborator.

8.5.2 *WristWiggle*

The *WristWiggle* module generates rotations at the wrist in order to allow *TipEstimate* to localize the tip. In the absence of tip prediction $\mathbf{x}_q$, *WristWiggle* randomly explores the wrist workspace. The 2 DOF of the wrist are ramped smoothly to random positions in the range of ± 60 degrees. This is often inefficient because the tip may go out of the field-of-view or be obstructed by the hand. Also, if the tip moves too quickly between two images, it may move outside the search window used in the visual block-matching algorithm. Significant motion blur may occur at the tip as well.

However, if $\mathbf{x}_q$ is known, then as we will see, the *TipPose* module can be used to keep the tip within the field-of-view. Knowing $\mathbf{x}_q$ also allows *WristWiggle* to restrict the tip velocity. If the arm and wrist move between configurations c_1 and c_2 in time t, then the tip

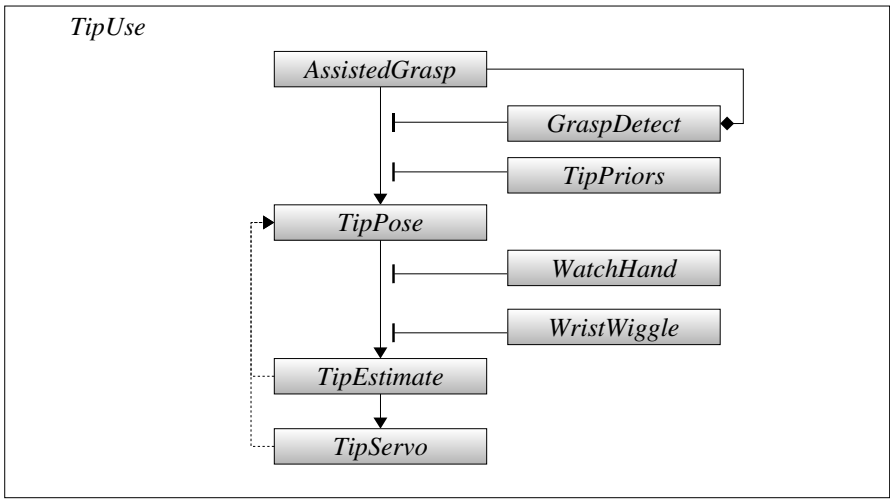


Figure 8-15: The *TipUse* module allows the robot to take an object from a person, quickly find its tip, and control the tip for a task.

velocity in the image is simply

$$v = \frac{\|T_{c_2}(\mathbf{x}_q) - T_{c_1}(\mathbf{x}_q)\|}{t},$$

where T_{ci} is the hand-to-eye transform. It is then straightforward for *WristWiggle* to interpolate between c_1 and c_2 such that $v < \epsilon_w$ for some limit ϵ_w.

8.5.3 *TipEstimate*

The *TipEstimate* module simply computes $\widehat{\mathbf{x}_t}$ using Equation 8.5. Typically, this is accomplished using approximately $200 - 300$ detections collected over approximately $10 - 15$ seconds. However, if the task category is defined in *TipPriors*, then *TipEstimate* replaces the uniform prior $p(\mathbf{x}_t)$ with $p(\mathbf{x}_t|q)$ in the estimation process. This allows increases the efficiency and robustness of the process and only 50 detections are used.

8.5.4 *TipUse*

The *TipUse* module enables Domo to take an object from a person, quickly find its distal tip, and control the tip for a task. The module that activates *TipUse* defines the task, including the task category and the control trajectory. The algorithm is as follows:

1. Secure a power grasp on an object relevant to the task.

2. Predict the location of the object's tip using learned priors.

3. Pose the manipulator so the predicted tip is visible by the camera.

4. Direct the eye gaze at the hand and generate rotational motion at the wrist.

5. Detect the tip through visual motion and estimate its true location.

6. Visually servo this tip through the desired tip trajectory.

As shown in Figure 8-15, *TipUse* integrates many of the modules we have discussed. To begin, Domo acquires an object through *AssistedGrasp* and predicts the tip location, $\mathbf{x}_q$, with *TipPriors*. Given $\mathbf{x}_q$, the *TipPose* module positions the manipulator so the tip is visible by the camera. Regardless of the wrist orientation, $\mathbf{x}_q$ should remain within the field-of-view if the hand is kept at an appropriate depth from the camera. We compute this depth as

$$z_c = \|\mathbf{x_q}\| \frac{f}{u},$$

for focal length f and field-of-view $2u$ pixels. The depth z_c uniquely defines a palm location $\mathbf{p}_d$ and tip location $\mathbf{x}_d$ that allows *TipPose* to point the object at the camera.

With the object pointing at the camera, *WatchHand* servos the gaze to the hand frame $\{H\}$. The arm and head maintain this posture but the wrist is allowed to move. *WristWiggle* generates small $\pm 30^\circ$ explorations about this pose, ensuring that the tip remains facing the camera. *TipEstimate* computes the tip location $\widehat{\mathbf{x}}_t$ using the tip prior. Finally, *TipServo* controls $\widehat{\mathbf{x}}_t$ through a trajectory specifed by the higher-level task.

8.6 The Task Relevant Hand

The robot's hand is certainly one of the most significant objects in the robot's environment. Perception and control of the hand can also be cast in the framework of *task relevant features*. The importance of a hand's feature is dependent on the task. In screwing a cap onto a bottle, we can consider the fingertip as the relevant feature. In flipping through papers, the tactile slip between the fingers is of primary importance. In picking up a soda bottle, the contact surface of the palm is critical. In this section we consider this last example. We present the *PalmServo* module which visually servos the contact surface of

Figure 8-16: Left: The robot's view during execution of *Palm*Servo. This occurs in an everyday environment which includes a shelf, a person, as well as natural lighting and a cluttered background. The green circles mark the convex shape of the open hand which is being servoed. Right: The motion edges used in the detection of the palm.

the open hand as it is brought to an object for grasping. This module is an extension of the *TipServo* module presented earlier.

8.6.1 *PalmServo*

When precise hand-eye calibration and a 3D model of the hand is not available, it can be necessary to visually detect the hand in the imge. The *PalmServo* module uses motion cues to detect the hand. This does not rely on a detailed model of the hand and is therefore extensible to a wide range of robot hands. Many researchers have created related methods for visual hand detection through motion. Fitzpatrick and Metta [42, 93] used image differencing to detect ballistic motion and optic-flow to detect periodic motion of the robot hand. For the case of image differencing they also detected the tip of the hand by selecting the motion pixel closest to the top of the image. Natale [97] applied image differencing for detection of periodic hand motion with a known frequency, while Arsenio [6] used the periodic motion of tracked points. Michel et. al. used image differencing to find motion that is coincident with the robot's body motion [94]. These methods localize the hand or arm, but do not select a particular feature on the manipulator in a robust way.

PalmServo is an extension of the *TipServo* controller. As shown in Figure 8-16, it visually servos the contact surface of the palm when the hand is open. The palm is defined by the convex shape of the open hand projected into the image. The open hand is always

directed towards the camera during this servoing, increasing the likelihood that the palm feature is visible. Visual detections of the feature are passed to the *TipServo* controller as a tip detection. *TipServo* then controls the point $\mathbf{x}_p$ in the image, where $\mathbf{x}_p$ is the kinematically estimated location of the palm feature in the hand frame.

PalmServo uses motion cues to detect the palm feature. The feature is detected using the *InterestRegions* module of Section xxx. However, we incorporate the hand motion prediction from Section yyy to increase the salience of moving hand edges. When used with the interest point operator, we can select for convex edges on the moving hand. The robot's kinematic model also provides a rough prediction as to the scale and location of the palm feature in the image. This is used to ignore detections that differ drastically in scale location from the prediction. Now, as the open hand moves in the image, *PalmServo* passes detections of the palm to the *TipServo* controller. We present results for *PalmServo* as part Section 9.1 which describes the *SwitchHands* module.

8.7 Discussion

In practice, not as modular as presented!!! TipServo requires careful consideration of motion so get tip, etc...

Discuss other featuers: shelf, surface, container opening, gravity, handles, ...

–do trials with estimation and with and coded. x objects. success and doing transfer. mean error while tracking vs hand annotated.

Placing on shelf: basic stabilty geometry/force analysis

–3D stereo integration, also pose,not just position is important

Chapter 9

Putting It All Together

In this chapter we demonstrate how the modules presented thus far can be integrated into useful, cooperative manipulation tasks.

9.1 *SwitchHands*

Transferring an object from one hand to another is a fundamental component of a person's manipulation repertoire. For example, a person taking notes while on the phone will transfer the phone to their non-writing hand. A person putting away dishes will remove an plate from the dishwasher with one hand, pass it to the other, and then place it in a cabinet. These seemingly simple acts increase a person's versatility while expanding the workspace of reachable locations. The *SwitchHands* module, shown in Figure 9-1, allows Domo to pass a grasped, cylindrical object from one hand to another. The algorithm is as follows:

1. Acquire an object and estimate its tip.
2. Brings both hands into the visual field-of-view.
3. Visually servo the palm of the empty hand to be just below the object's tip.
4. Lower the empty hand through force control until the other hand detects contact.
5. Form a power grasp and detect its success.
6. If no success, back out and retry the servo.
7. If both hands are grasping, release the older grasp.

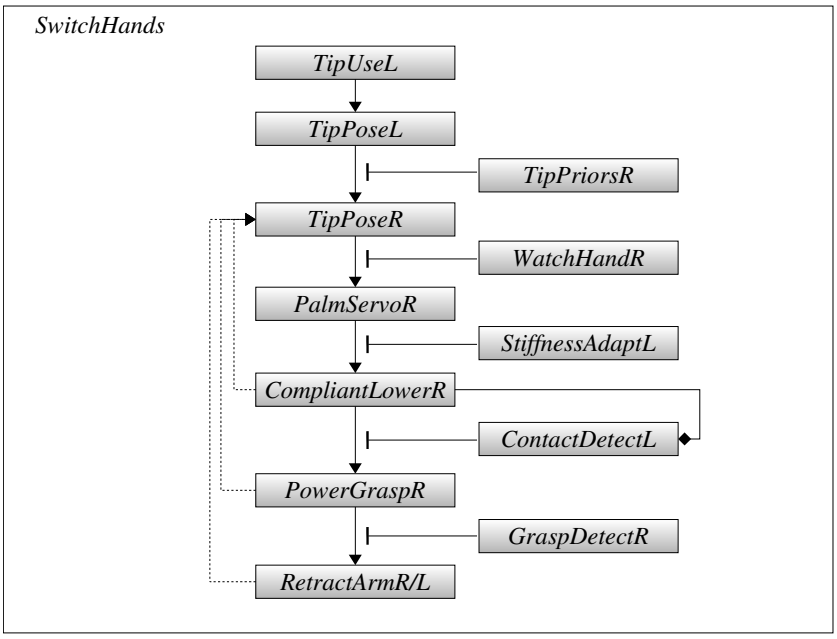


Figure 9-1: The *SwitchHands* module to transfer an object from the left hand to the right.

Lets consider the case where an object is passed from the left hand to the right. *SwitchHands* begins by activating *TipUseL* in order to acquire an object and find its distal tip. Instead of servoing the object, *TipPoseL* places the object at a canonical location and points the object's primary axis at the camera. Next, *TipPoseR* brings the right hand into a canonical location a set distance from the left hand. The top of the hand also points towards the camera. This configuration of the manipulators is shown in Figure 9-7-E. *WatchHandR* keeps the hand within the field-of-view as *PalmServoR* visually servos the palm of the hand to the grasped object. If the object tip is at $[x_c, y_c, z_c]^T$ in the camera frame, the target location for the right palm is just below the tip at $[x_c, y_c, z_c + \Delta z]^T$ for offset Δz. When the palm has been servoed into place, the *StiffnessAdaptL* makes the left arm compliant while *CompliantLowerR* lowers the right hand onto the object. The displacement of the left hand is detected by *ContactDetectL* and the right hand forms a power grasp on the object. If *GraspDetectR* is signaled, the left hand release the object and both arms are retracted. If a grasp is not made or contact is not detected, *SwitchHands* reattempts the process. Figure 9-7 shows the execution of *SwitchHands*.

9.1.1 Results

It would seem that *SwitchHands* could be easily implemented using open-loop control given enough assumptions about the object and its pose in the hand. However, even for cylindrical objects, this controller would not be robust to variation of the cylinder's pose in the hand. Also, if the grasp aperture of the open hand is close to the size of the cylinder, precise calibration of the arms is required. *SwitchHands* improves the open-loop controller through visual control of the palm and estimation of the object's pose in the grasp. We experimentally tested this robustness by xxx.

9.2 *BimanualFixture*

In a bimanual fixturing task, a robot holds an object between its two hands in order to assist a person. For example, a robot could hold a laundry basket while the person loads in the clothing, or a robot could hold a serving tray at a dinner table. In this section we describe how Domo can control the position and orientation of a bimanually held box in order to assist a person loading objects in to it.

The general problem of controlling the internal forces applied to a bimanually grasped object is well understood theoretically [142]. These controllers typically assume that a full, 6 DOF force wrench can be applied instantaneously to the rigidly held object. This would be difficult in practice given Domo's actuators and kinematic limitations. In addition, it is not possible to assume a rigid, point contact grasp given Domo's compliant skin on the hand. Instead, we consider a simplified form of the problem and leverage the robot's inherent compliance to maintain stable control of the box. A non-prehensile grip is used such that the object is squeezed between the palms of the hands and the palms are aligned to the box surface. The size of the box is unknown but assumed to be of a reasonable size and shape for the manipulators to achieve the task.

The *BimanualFixture* algorithm, illustrated in Figure 9-3, is as follows:

1. Upon request, raise both arms to cue the person to hand the box.
2. Detect placement of the box between hands through contact forces.
3. Use virtual spring controllers and low arm stiffness to stably grasp the box between palms.

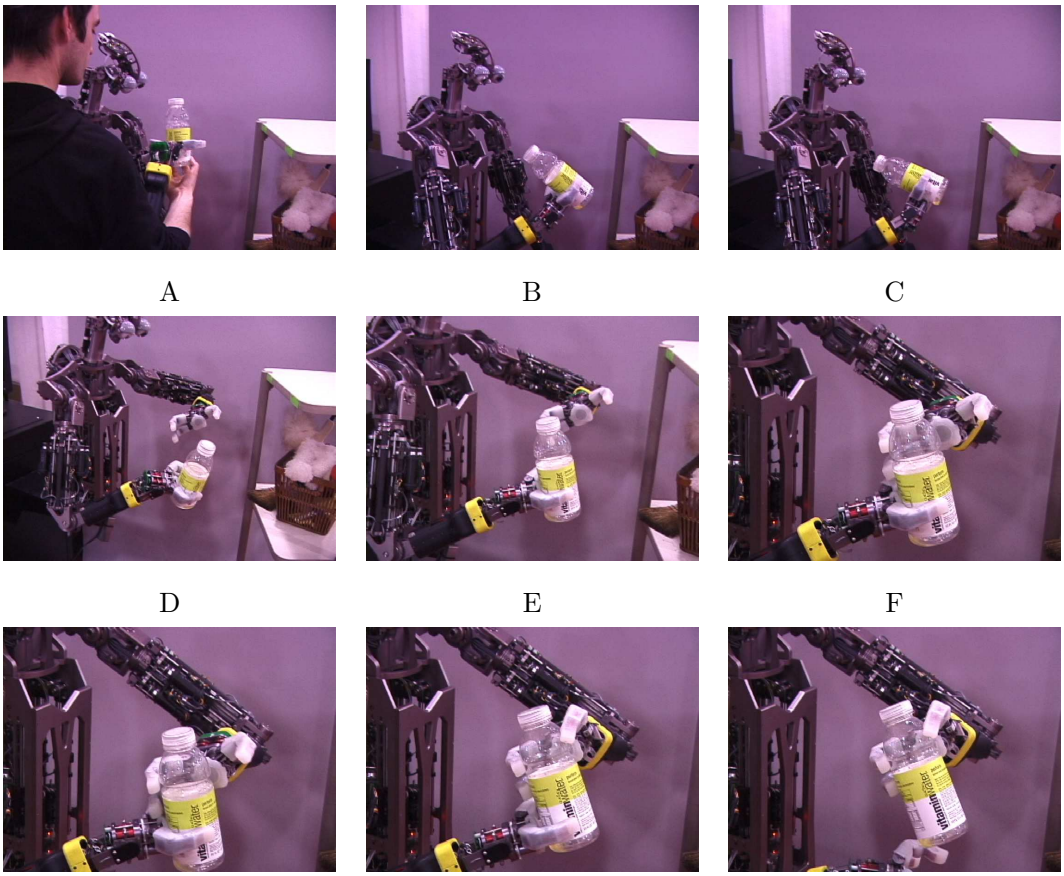


Figure 9-2: Video stills of the *SwitchHands* module. (A) A bottle is grasped using *AssistedGrasp*. (B-C) The tip of the bottle is estimated. (D) The arms are placed in canonical poses. (E) The top of the hand and the bottle tip are pointed to the camera. (F-G) *PalmServo* brings the palm surface to the bottle. (H)The palm pushes down on the bottle *ContactDetect* signals the hand to close. (I) The other hand releases its grasp.

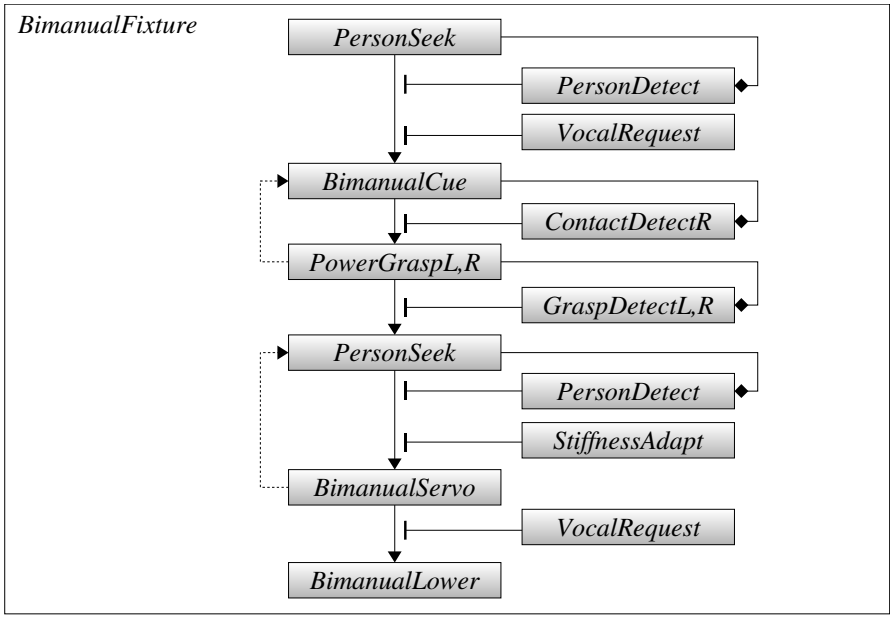


Figure 9-3: The *BimanualFixture* module. When both hands are empty (*EmptyA,B*), the robot seeks visual detection of a person. Upon seeing a person and a vocal request, the robot reaches with both arms to cue (*BimanualCue*) the person to hand it a box to hold. *BimanualGrasp* forms a stable grasp on the box after manipulator contact is made. As the person loads objects into the box, *FixtureServo* maintains a constant squeezing force on the box while positioning it near the person. Finally, *BimanualPlace* lowers the box on to a nearby table upon request. (Right, *FixtureServo*) The *PersonSeek* module controls the head to increase the likelihood of *PersonDetect* through visual search and tracking. When a stable grasp is made on the object (*GraspDetectA,B*) and a person is present, the servo loop updates the arm pose *(PoseA,B)* to maintain the grasp and bring the box near the person. *StiffnessAdapt* lowers the wrist and arm stiffness so that the arms behave as soft virtual springs squeezing the object.

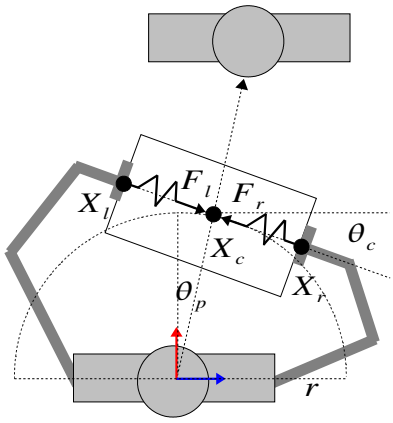


Figure 9-4: Collaborative bimanual fixturing as described in the text.

4. Visually the track person as they load items into the box and reposition it near them.

5. Upon request, place the box on a nearby table or shelf if it is present.

In the next section we describe the virtual spring controllers for performing the bimanual grasp.

9.2.1 Control

We consider the case as shown in Figure 9-4 where the box is already stably grasped. From the robot's kinematic model, we know that the palms are at X_l and X_r. We define $X_c = \frac{X_r + X_l}{2}$ as the object's estimated center of mass (COM). The controllers for the two arms are to keep X_c at a fixed radius r from the body, at a height z_c, and in the gaze direction, θ_p, towards a person in the environment. They also keep the orientation of the box, defined by the ray $X_r - X_l$, at a desired angle, θ_c, to the body. It is assumed that the top of the box is kept normal to gravity for the task duration.

In our simplified model the manipulators impart only forces but no moments at the palms due to the non-prehensile grasp. It can be shown [142] that the resultant forces F_b, moments M_b, and interaction forces T_b due to the applied forces F_l and F_r are

$$\begin{bmatrix} F_b \\ M_b \\ T_b \end{bmatrix} = G \begin{bmatrix} F_l \\ F_r \end{bmatrix},$$

where G is the bimanual grasp description matrix

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{d}{2} & 0 & 0 & -\frac{d}{2} \\ 0 & -\frac{d}{2} & 0 & 0 & \frac{d}{2} & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and $d = \|X_r - X_l\|$. The equillibrating forces, F_b and M_b, maintain equillibrium under an external load such as gravity. The interaction forces, T_b, are internal to the grasped object. In our case, T_b is the squeezing force on the box. Its magnitude, $\|T_b\|$, should be sufficently large such that the the contact friction with the palm is sufficient to support the weight of the box.

Virtual spring controllers create T_b by applying opposing forces along ray $(X_r - X_l)$. These are defined by $F_r = -k_f(X_r - X_c)$ and $F_l = -k_f(X_l - X_c)$. The spring stiffness, k_f, is then proportional to $\|T_b\|$. Rather than controlling the force at the palm directly, we approximate this force through a constant error in the desired joint angle of the arm. For the right arm, this is $\Delta\theta_r = \sigma J^T F_r$ given scalar σ. This approximation takes advantage of the joint angle controller of Equation XXX, where the controller error is linearly related to joint torque. By keeping the stiffness of the arms relatively low and using this method, the arms remain can remain in stable contact with the box under large disturbances.

In practice, we do not know the true pose of the box between the robot's hands. The forces applied to it are a complex function of the geometry of the box, the palm, and the joint torques of the arm. Rather than attempt to fit the world to our simple model, we allow the compliance in the arms and hands, combined with the large contact surface of the palm, to compensate for non-zero moments on the box. A 2 DOF wrist controller is used to keep the surface normal of the palm aligned with the ray $(X_r - X_l)$. However, the stiffness

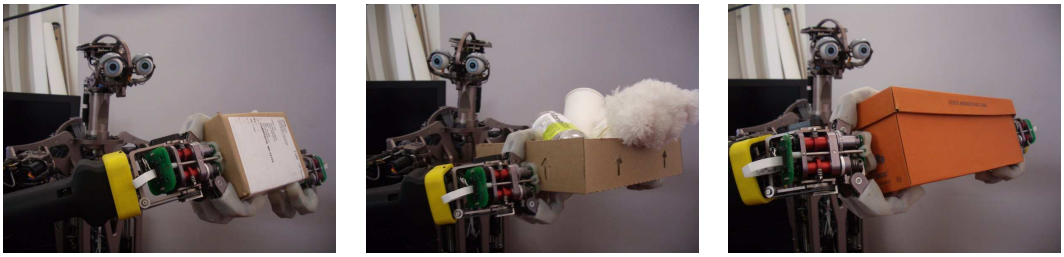


Figure 9-5: The virtual spring controller is able to maintain a stable grasp on boxes of varying sizes. For each of the three boxes tested, a person applied external disturbances to the COM of up to 0.2m . The controller was able to maintain the grasp and restore the box position.

of this controller is kept low. This, combined with the compliant skin of on the palm, allows the box and wrists to passively realign themselves such that contact is maintained with the box and the resultant wrench is minimized.

As the person moves around the room, the gaze direction towards the person changes from θ_p to $\theta_p^{'}$. Given r, this defines a new target COM of $X_c^{'} = [rsin(\theta_p^{'}), rcos(\theta_p^{'}), z_c]$. We define the desired box orientation, $\theta_c = \theta_p^{'}$, such that the box remains tangent to the arc of radius r. We also infer the box width as $d = \|X_r - X_l\|$, yielding a new target location for the right palm $X_r^{'} = X_c^{'} + [\frac{d}{2}cos(\theta_p^{'}), \frac{d}{2}sin(\theta_p^{'}), 0]$. The desired joint angle of the right arm is computed as $\theta_r = IK(X_r^{'}) + \Delta\theta_r$, where $IK(\cdot)$ is the inverse kinematic function. A similar update is computed for the left arm. For a smoothly changing θ_p , this has the desired affect of incrementally adjusting the box COM and orientation while maintaing a constant squeezing force on it. Finally, the height of the box can be controlled within the joint limitations of the manipulators by varying z_c.

9.2.2 Results

We first tested the ability of the virtual spring to maintain a stable grasp on boxes of varying sizes. As shown in Figure 9-5, three boxes of different sizes were successfully tested. For each, the box was grasped and servoed to a fixed position for 8-10 seconds. A person applied external disturbances of the COM of up to $0.2m$ while the controller maintained its grasp and restored the box position. Figure 9-6 shows the controller response for one of the boxes.

We then demonstrated the complete *BimanualFixture* module. As shown in Figure 9-7,

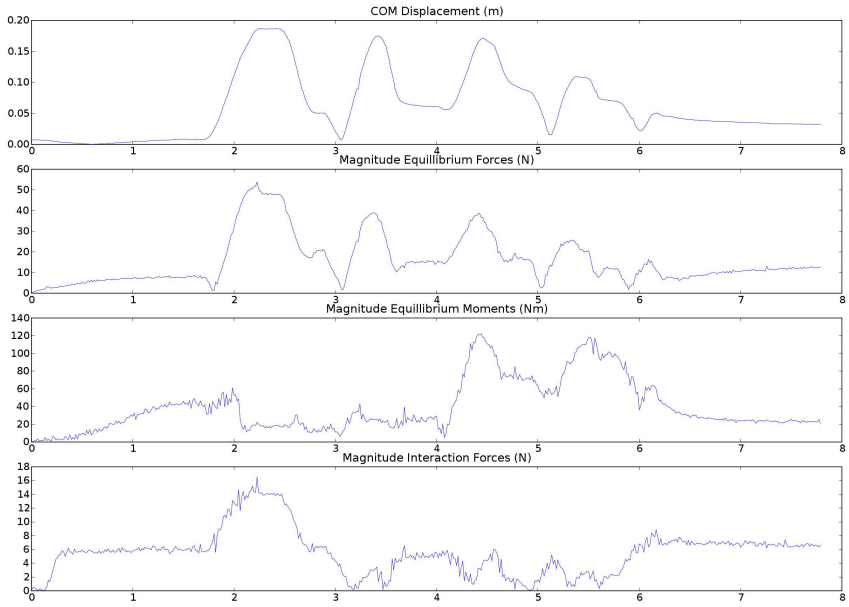


Figure 9-6: The magnitude of the equillibriating and interaction forces to a disturbance during bimanual fixturing. A box was grasped and servoed to a fixed position for 8 seconds. (Top) During this time a person displaced the box COM from its rest position by up to 0.2m . (Middle) The controller responded with equillibriating moments ($\|M_b\|$) and forces ($\|F_b\|$) to restore its position. (Bottom). Interaction forces ($\|T_b\|$) were maintained throughout the experiment despite the disturbances. Although they momentarily approach zero in several places, a stable grasp is maintained. This is likely due to adaptation of the fingers and wrist to the displaced surface.

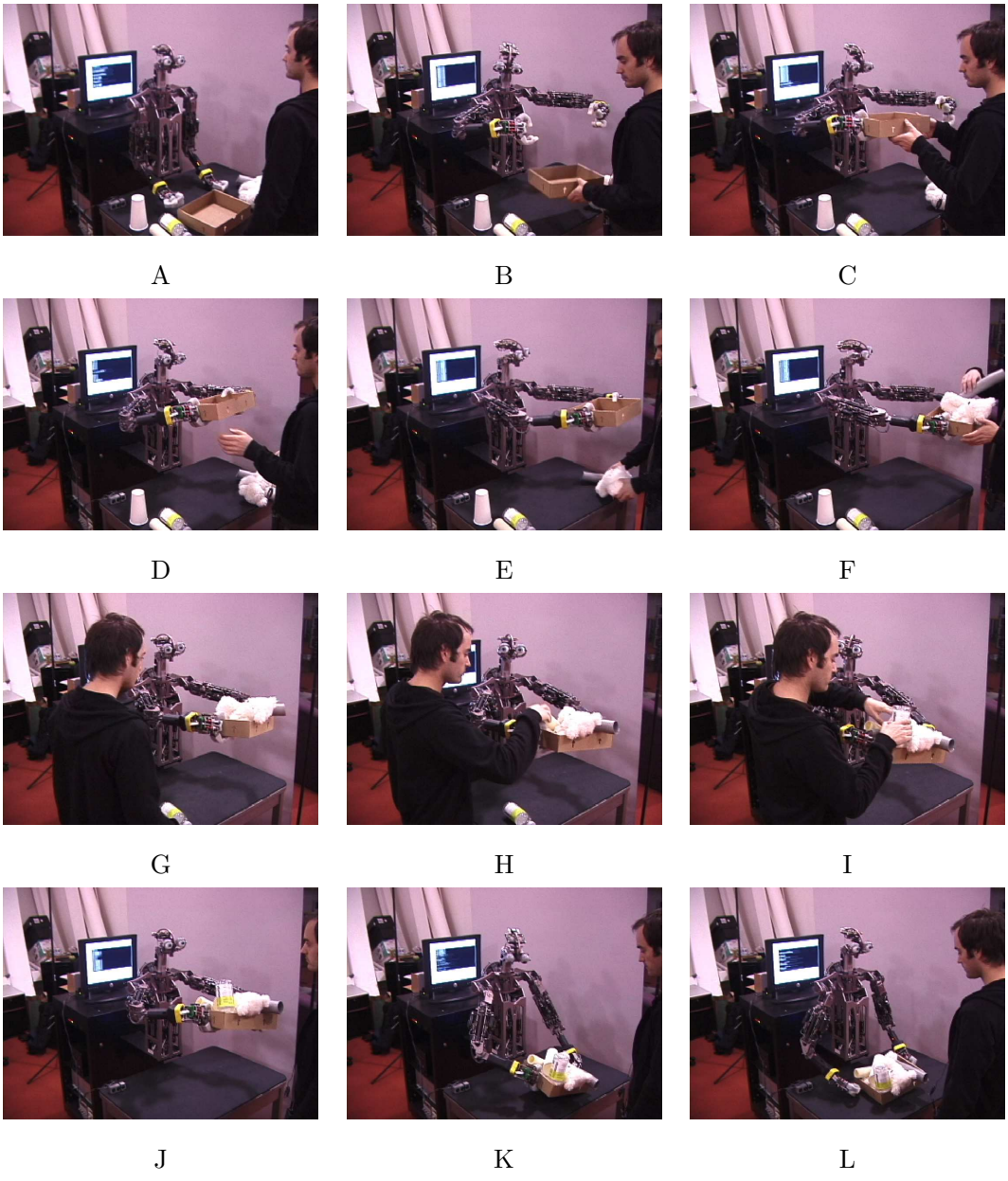


Figure 9-7: Domo assists a person in putting things away.

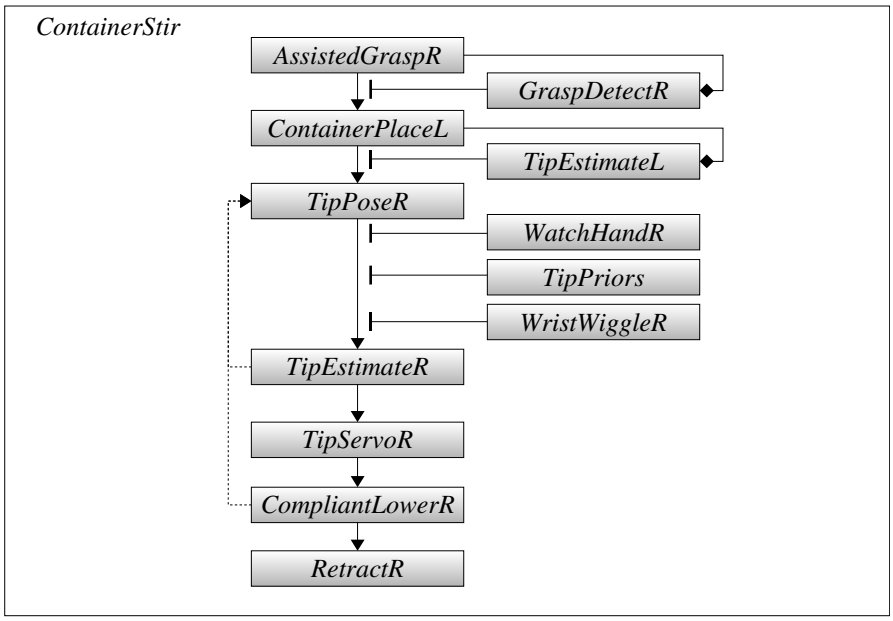


Figure 9-8: The *ContainerStir* module as described in the text.

Domo is able to assist a person in putting away objects. The module is also able to fail gracefully in case the box is removed from its grasp or the person is no longer in view.

9.3 *ContainerStir*

In the *ContainerStir* module, Domo grasps a stirring spoon and a container such as cup or coffee mug. It inserts the spoon into the mug and stirs the contents. This involves visually guided bimanual insertion of two objects. This type of peg-in-hole insertion is a well studied area, especially in industrial robotics. XXXCite: Inoue, etc, peg align, ...XXX. The performance of these systems is quite good, even under moderate uncertainty in the object pose. In one of the first examples of using compensatory actions, Inoue demonstrated that holding the peg at angle increases task success under pose uncertainty. However, assumptions about the shape and location of the two objects are quite strong. Also, most work on insertion is not bimanual. The bimanual task can be both easier and harder. The ability to control the pose of the second object allows for greater versatility and sensing during task execution. However, it also increases the uncertainty of the relative pose between the two objects. A compliant grasp by the second manipulator is also helpful,

allowing the pose of the object to adapt to the first.

The *ContainerStir* algorithm, illustrated in Figure 9-8, is as follows:

1. Grasp a spoon and container with *AssistedGrasp*.
2. Lower the container onto a table and detect its insertion opening with *ContainerPlace* (see next section).
3. Bring the tip of the spoon approximately above the opening opening using the *TipPriors* for insertion.
4. Fixate the gaze on the hand while generating spoon motion at the wrist.
5. Estimate the location of the spoon tip using *TipPriors* and *TipEstimate.*
6. Visually align the tip to the center of the opening with *TipServo*.
7. Compliantly lower the spoon into the container, perform a simple stirring motion, and retract.

ContainerStir is primarily a composition of other modules. The critical step is the localization and servoing of the spoon tip. We adopt an approach similar to Inoue where the *Pose* module aligns the spoon at a 45 degree angle to the table. This prevents visual obstruction of the tip by the hand and expands the range of acceptible misalignment when performing the insertion. During *TipServo,* the tip is kept on the visual ray to the center of the container opening. The depth of the tip is then increased along the ray until the tip is just above the insertion location. Like the RCCCXXX, the manipulator's wrist stiffness is kept low while performing the insertion, allowing for greater misalignment. Although insertion success is not directly detected through the interaction forces, large hand displacements during *CompliantLower* can signal the module to restart the process. The algorithm for *ContainerStir* could be applied with little modification to other insertion-type tasks such as pouring a bottle of water. However, kinematic limitations in the manipulator wrist make this difficult.

9.3.1 *ContainerPlace*

ContainerPlace assists *ContainerStir* by reducing the location uncertainty of a container's insertion opening. The algorithm, illustrated in Figure 9-9, is as follows:

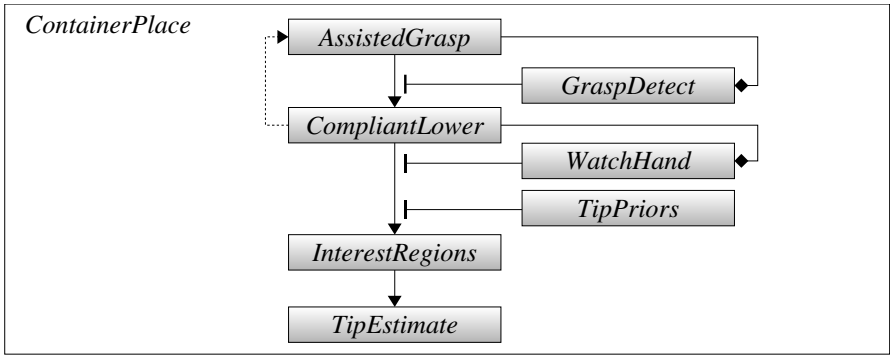


Figure 9-9: The *ContainerPlace* module as described in the text.

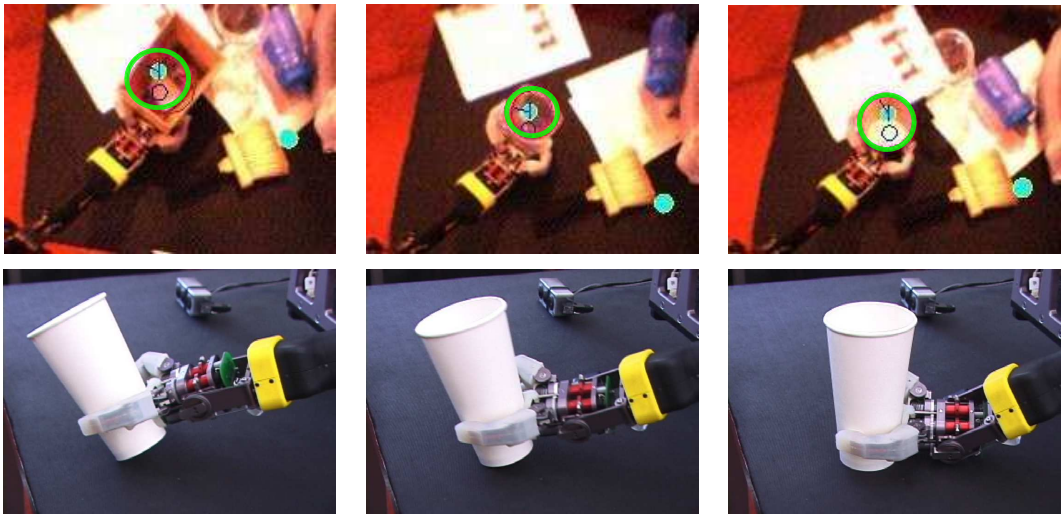


Figure 9-10: Execution of the *ContainerPlace* module. **Top:** The *InterestRegions* module find the roughly circular opening of a box, jar, and a bowl. The detector is farily robust to cluttered backgrounds. **Bottom**: Image sequences of **CompliantLower** using the table to align a grasped cup.

1. The container is grasped by *AssistedGrasp*.

2. *CompliantLower* brings the object down onto a table with low arm stiffness, allowing it to align with the surface.

3. The hand is fixated by *WatchHand* while *InterestRegions* visually detects the container opening near the location predicted by *TipPriors*.

4. *TipEstimate* computes the likely opening location in the hand frame.

Flat surfaces are common in human environments, and like the *SurfacePlace* module, *ContainerPlace* takes advantage of a flat surface to align the object. This is shown in Figure 9-10. Also, the table is used as a stable support during insertion, much like a person resting their cup on a table before pouring a cup of coffee. We assume that a table exists, that the container can be grasped with one hand, and that it posesses a roughly circular opening for insertion. The insertion opening is a task-relvant feature, inclusive of a variety of objects such as drinking glasses, bowls, small boxes, and coffee mugs. This feature is detected using the *InterestRegions* module. However, the module is configured to ignore motion cues and to find interest regions in single images. It is localized near the expected opening location provide by *TipPriors*. *TipEstimate* computes the likely tip location using this xxx single feature detection. Figure 9-10 shows opening detections on a variety of objects. It is worth noting that the tip prior allows the detection to generally works well even on cluttered tables. (XXX static???)

9.3.2 Results

The *ContainerStir* can generalize across insertion object and containers. We tested, seven trials each, inserting a mixing spoon, water bottle, paint roller, and paint brush into a paper cup. We also tested, seven trials each, inserting the mixing spoon into a papercup, bowl, coffee mug, and jar. Each trial took less than 20 seconds to complete and was done over a visually cluttered table. The grasp on the insertion tool was varied in each trial. The grasp orientation varied by approximately $\pm 20^{\circ}$ along the axis of the hand's power grip. The grasp height on tool handle was varied by approximately $\pm 50mm$. However, extreme variations were not attempted because of their unlikelihood given the tip prior. In a successful trial, Domo fully inserted the tool into the container. The diameter of the

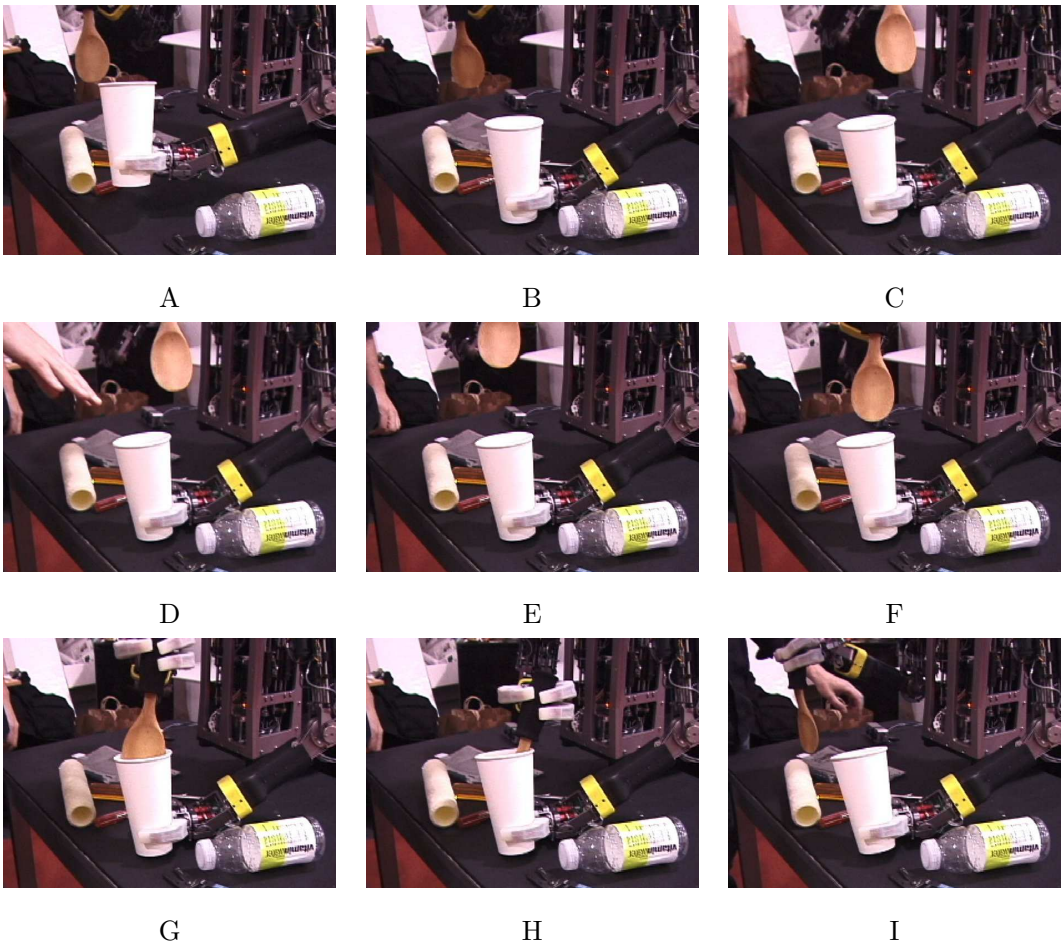


Figure 9-11: Execution of the *ContainerStir* module with a cluttered background. (A-B) *ContainerPlace* aligns the cup to the table and detects its opening. (C) Using *TipPriors,* the spoon is brought over the cup. (D) The wrist wiggles the spoon and *TipEstimate* estimates the true spoon tip (despite background motion from a person's hand). (E-G) Using the estimated tip, the spoon is brought over the cup by *TipServo* and compliantly lowered in. (H-I) Stirring motion is executed and the spoon is removed.

	Papercup	Bowl	Box	Coffee mug	Jar
Spoon	7/7	7/7	7/7	6/7	7/7
Water bottle	6/7				
Paint brush	6/7				
Paint roller	5/7				
Spoon-prior	1/7				

Figure 9-12: Task success for *ContainerInsert.* In a successful trial, Domo inserted the tool (rows) into the container (columns). For comparison, the Spoon-prior trials were done without visual restimation of the tool tip and instead used the expected location.

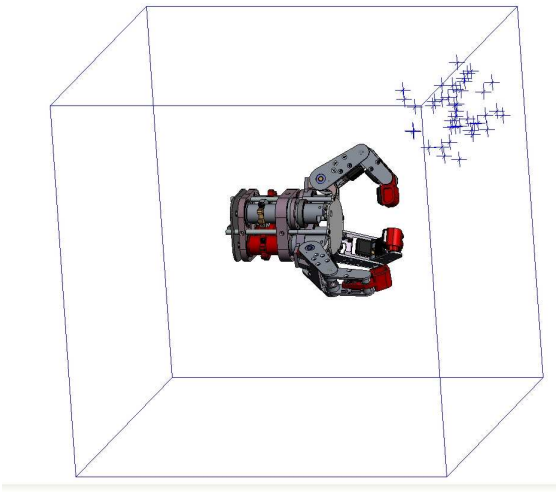


Figure 9-13: The likely tip locations, as computed by *TipEstimate,* for each trial of the *ContainerInsert* experiment.

insertion opening varied between $75 - 100mm$ across the containers while the tip diameter varied between $40 - 60mm$. The results of these trials are shown in Figure 9-12.

As the results show, *ContainerInsert* was successful in roughly 90% of the trials. When the visual restimation of the tip was disabled, the success rate fell to about 15%. However, the tip locations, $\widehat{\mathbf{x}}_t$, do not span the full range of locations one might expect for people. The distribution of $\widehat{\mathbf{x}}_t$ over these trials are shown in Figure 9-13. We would expect the algorithm to generalize to a wide range of locations. However, for *TipEstimate* to perform as expected, the tip must remain in the field-of-view, the hand must not obstruct the view, and the velocity of the tip must not exceed the limits of the visual motion model. These constraints require tuning of the *WristWiggle* module, which makes small random motions of the wrist around a setpoint. If *WristWiggle* adapted its setpoint and the amplitude of its motion based on feedback from *TipEstimate* and *TipPriors*, then the generality of *ContainerInsert* could be expanded further. Also, *ContainerInsert* could be extended to incorporate force-feedback during the insertion. The 3D orientation of the tip could also be sense using stereo camera. We presently assume that the tip is aligned with the object body, but this is not always the case.

9.4 *PuttingStuffAway*

A significant portion of domestic and workplace tasks involve putting stuff away. Examples include unloading a dishwasher, putting groceries in the refigerator, clearing a desk, stocking items in a supermarket, and cleaning a bedroom. A robot that can autonomously put stuff away in all of these varied settings would certainly be useful. However, an intermediate step is for a robot to assist a person in these tasks. For an individual with serious physical limitations, this help might allow the person to maintain autonomy in everyday activities that would otherwise require help from another person. For example, an elderly person in a wheelchair might use a robot to put a book back on a shelf.

To this end,the *PuttingStuffAway* enables Domo to take items from a person and put them on a shelf. The algorithm is as follows:

1. Locate a useable shelf surface.

2. Upon request, take an item from a person.

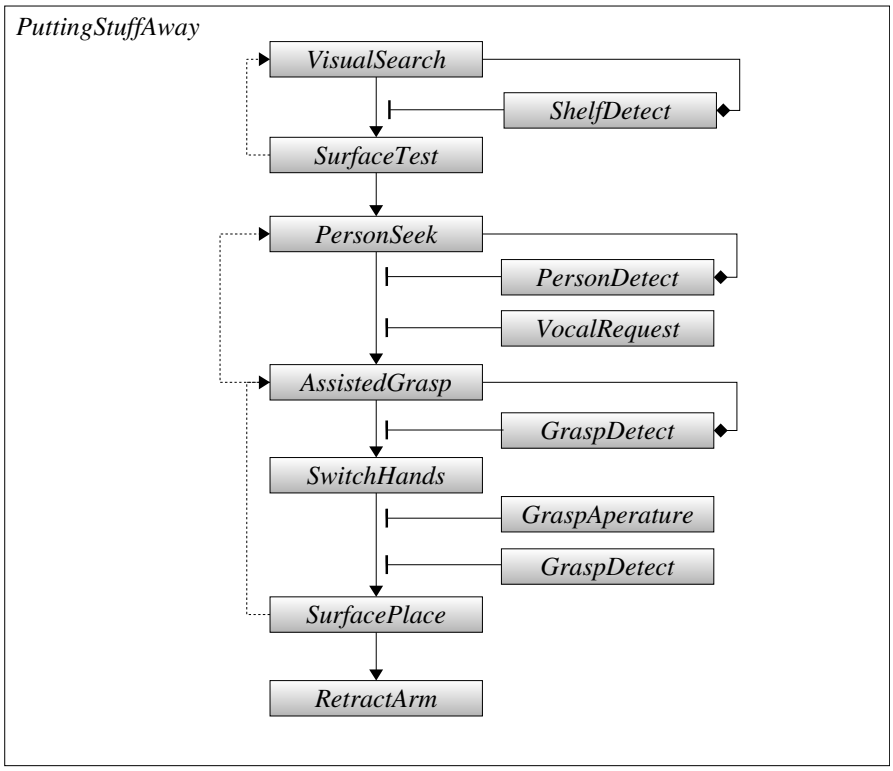


Figure 9-14: The *PuttingStuffAway* module described in the text.

3. Transfer the object to the hand closest to the shelf if necessary.

4. Place the item on the shelf.

As shown in Figure 9-14, *PuttingStuffAway* integrates the *AssistedGrasp*, *SurfaceTest*, *SwitchHands,* and *SurfacePlace* modules into a single, high-level task. The task can be decomposed into a shared set of perceptual and motor modules shown in Figures 6-5,7-8,9-1,and 6-8. To begin, *PuttingStuffAway* attempts to locate a useable shelf surface with *SurfaceTest*. If it has physically located a shelf but the shelf moves within the SES, then *SurfaceTest* will reestimate its location. Next, the robot awaits its collaborator to request assistance. When *PersonDetect* and *VocalRequest* are signaled, *AssistedGrasp* takes an object that the collaborator hands it. *AssistedGrasp* uses whichever arm is closest to the person. *PuttingStuffAway* determines which hand is closest to the shelf. If necessary, it uses *SwitchHands* to transfer the object to the hand nearest the shelf. If *GraspDetect* signals that *SwitchHands* is successful, then *SurfacePlace* puts the item on the shelf. The item

is placed upright or horizontally depending on the size of the items base, as estimated by *GraspAperture*.

9.4.1 Results

We tested *PuttingStuffAway* in 18 trials with two subjects, where each trial lasted approximately one minute. A trial consisted of the subject handing Domo a bottle, Domo transferring the bottle to its other hand and then placing it on the shelf. One trial is depicted in Figure 9-15. Each subject performed 3 trials on each of the 3 bottles shown in Figure 9-16. The bottles vary in diameter from $40 - 75$mm and length from $100 - 200$mm. For each subject, the shelf remained stationary and the *SurfaceTest* module executed only once at the start of the experiment. We measured success using the following criteria:

1. Grasp: Stable grasp after transfer of the bottle from the person to the robot.
2. Switch: Stable grasp after transfer of the bottle between hands.
3. Place: Bottle X was left on the shelf.
4. Stand: Bottle X was left on the shelf standing upright.

As seen in Figure 9-17, Domo was largely successful at the task for the given objects. One subject was experienced in working with the robot at this task and consequently achieved a higher success rate. Failures were typically a result of insecure grasps being formed during the object transfer phase. Variability in the subject's placement of the object in the robot's hand tended to be amplified by the transfer operation.

9.4.2 Discussion

PuttingStuffAway effectively extends the collaborator's reach, allowing her to place objects in locations that might be difficult or uncomfortable to access without assistance. If this skill were combined with a mobile base, the person's effective reach could be dramatically extended. While the module demonstrates end-to-end execution of a useful, cooperative task, there are many ways in which it can be extended. The assumptions of *SurfacePlace* and *SwitchHands* restrict the types of items that can be placed. Richer sensing of the object's features could allow for robust placement of non-cylindrical items. As an example,

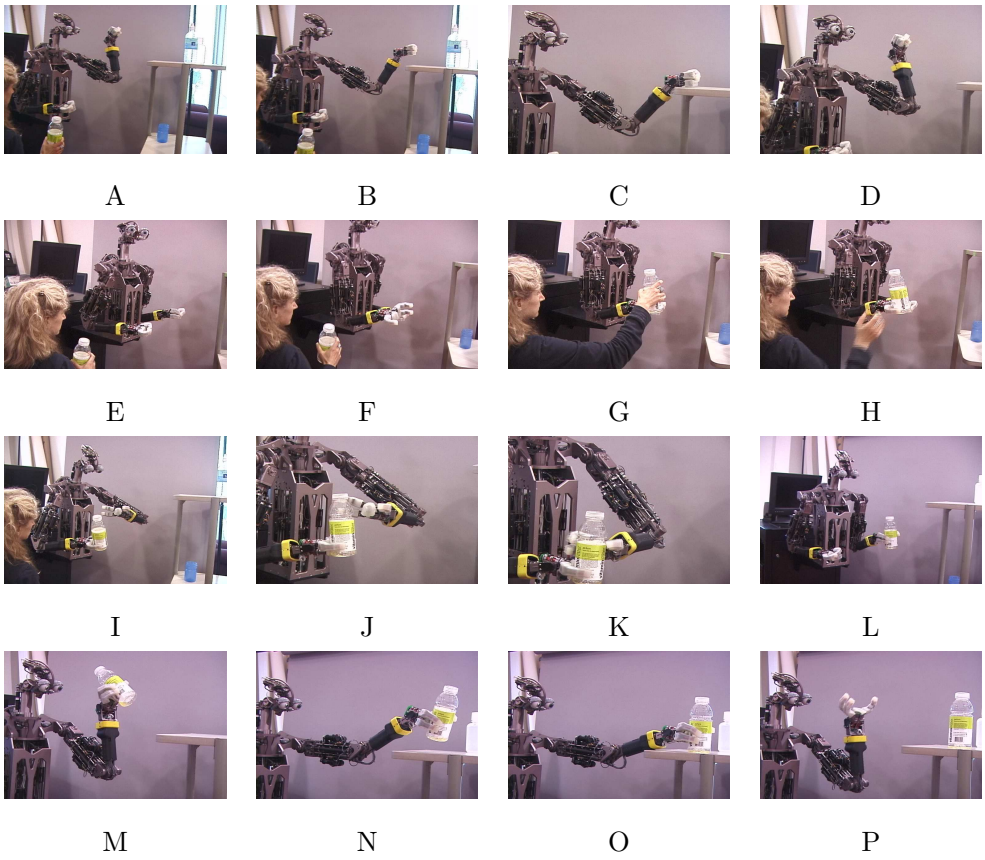


Figure 9-15: Execution of the *PuttingStuffAway* module. (A-D) **Hypothesis testing**: A shelf is rolled up to Domo. It is visually detected and *SurfaceTest* physically verifies its location. (E-H) **Cooperative interaction**: A person requests assistance from Domo. They are visually detected and *AssistedGrasp* cues the person to hand it the bottle. (I-L) **Expanding the workspace**: The shelf is out of the person's reachable workspace but within the workspace of the robot's left arm. *SwitchHands* transfers the bottle from the right to left hand. (M-P)**Placement**: Domo places the bottle on the shelf at the learned location. The manipulator compliance and downward force allow the bottle to become aligned with the shelf.

A B C

Figure 9-16: The three bottles used in our experiments.

	Grasp	Transfer	PlaceA	StandA	PlaceB	StandB	PlaceC	StandC
Subject 1	9/9	9/9	3/3	3/3	3/3	3/3	3/3	2/3
Subject 2	9/9	8/9	2/3	2/3	3/3	2/3	2/3	1/3

Figure 9-17: Experiment results for *PuttingStuffAway*, with 18 trials and two subjects.

Ashutosh/Ng grasp points cite XXX. Also, all items are placed at the same location on the shelf. Clearly, perception and planning of placement locations will be necessary component of real-world tasks.

9.5 *HelpWithChores*

HelpWithChores is a final demonstration of our approach that integrates all of the modules described thus far. *HelpWithChores* enables Domo to assist a person in tasks that might be expected of a robot working in a domestic setting.

As shown in Figure 9-18, *HelpWithChores* integrates the *BimanualFixture*, *Container-Insert*, and *PuttingStuffAway* among others. These modules run concurrently, allowing a collaborator to vocally request them at any time. A rich integration of these modules allows for a believable cooperative experience for the collaborator. If she sees that Domo is failing at a task, she can provide vocal (*VocalRequest*) or contact (*ContactDetect*) feedback to alert the robot. If Domo accidentally drops an object (*GraspDetect*), she can pick it up and ask the robot to grasp it again (*AssistedGrasp*). Alternatively, at anytime she can ask Domo to hand her a grasped object (*AssistedGive*).

This behavioral richness is central to the creature robot approach. It requires that a

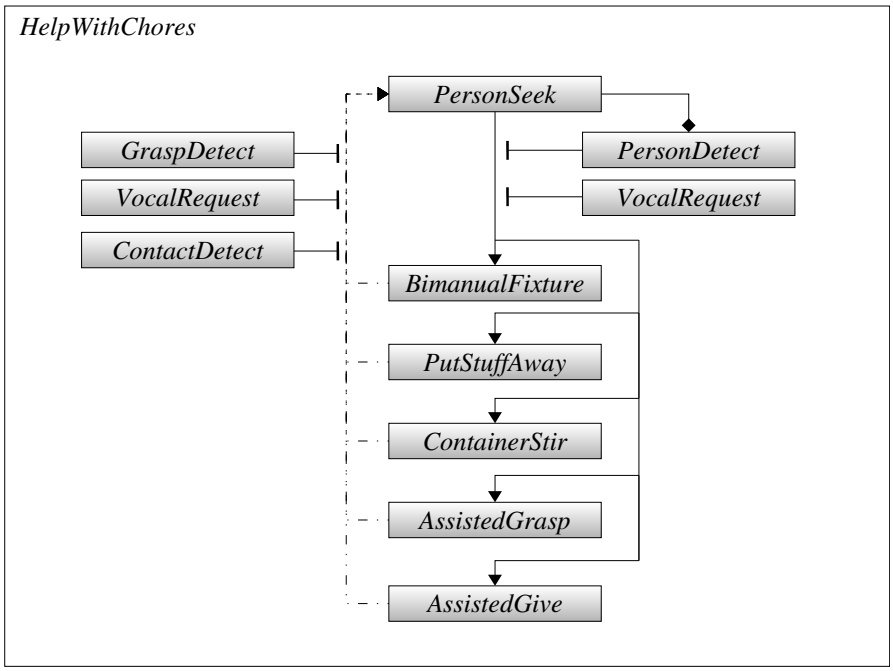


Figure 9-18: The *HelpWithChores* module described in the text.

module like *PuttingStuffAway* is not an algorithmic playback of a manipulation task. The collaborator should not serve as a proxy to a keyboard, making simple API calls into the code. The robot should at all times be responsive to detected failures during the task and redirection by the collaborator.

With this in mind, one possible *HelpWithChores* scenario is as follows:

1. Domo is positioned at a table cluttered with objects and near a shelf.
2. A person asks for help preparing meal by mixing food in a bowl. She hand's Domo a bowl and a spoon, it inserts the spoon into the bowl and stirs the contents.
3. She asks Domo for the bowl. Domo extends the grasped bowl to the her.
4. The person takes the "prepared" meal from Domo and asks it to put the spoon away on the shelf. Domo complies.
5. The person finishes her meal and asks Domo to take the bowl. Domo takes the bowl and at her request, passes the bowl to the other hand and puts it on the shelf as well.

A B C D

E F G H

I J K L

M N O P

Figure 9-19: Execution of the *HelpWithChores* module during one consecutive run.

6. Next, the person asks for help cleaning up. At her request, Domo holds a box for her while she clears the table into the box.

This scenario is realized by Domo and the author as one consecutive cooprative manipulation task. This is shown Figure 9-19. As we can see, even with the limited perceptual and motor abilities of Domo, a rich behavioral integration allows it to extend beyond simple, experimental demonstrations. We begin to see the potential for Domo to be a truly useful, partner robot.

9.6 Discussion

xxx

Chapter 10

Conclusions

10.1 Sumary of significant contributions

10.2 Biography

Appendix A

Module Summary

1. *AssistedGrasp*
2. *AssistedGive*
3. *BimanualCue*
4. *BimanualFixture*
5. *BimanualLower (CompliantLower???)*
6. *BimanualServo*
7. *CameraReach*
8. *CompliantLower*
9. *ContactDetect*
10. *ContainerPlace*
11. *ContainerStir*
12. *FixtureServo*
13. *GraspAperture*
14. *GraspDetect*
15. *GraspRelease*

16. *HelpWithChores*
17. *InterestRegions*
18. *PalmServo*
19. *PersonDetect*
20. *PersonReach*
21. *PersonSeek*
22. *PowerGrasp **
23. *PutStuffAway*
24. *RetractArm **
25. *StiffnessAdapt*
26. *ShelfDetect*
27. *SurfaceTest*
28. *SurfacePlace*
29. *SurfaceReach*
30. *SwitchHands*
31. *TipEstimate*
32. *TipPose*
33. *TipPriors*
34. *TipServo*
35. *TipUse*
36. *VisualFixate*
37. *VisualSearch*
38. *VocalRequest*

39. *WatchHand*

40. *WristWiggle*

Bibliography

[1] Bryan Adams, Cynthia Breazeal, Rodney Brooks, and Brian Scassellati. The Cog project. *IEEE Intelligent Systems*, July/August 2000. To appear.

[2] P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *In Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 268–272, 1987.

[3] J.S. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man and Cybernetics*, 21:473–509, May 1991.

[4] Ronald Arkin. *Behavior Based Robotics*. MIT Press, 1998.

[5] Ronald C. Arkin. Homeostatic control for a mobile robot: Dynamic replanning in hazardous environments. In William J Wolfe, editor, *SPIE Proceedings 1007, Mobile Robots III*, pages 407–413, 1989.

[6] A. Arsenio and P. Fitzpatrick. Exploiting cross-modal rhythm for robot perception of objects. In *Proceedings of the Second International Conference on Computational Intelligence, Robotics, and Autonomous Systems*, December 2003.

[7] Lijin Aryananda and Jeff Weber. Mertz: A quest for a robust and scalable active vision humanoid head robot. In *Proceedings, IEEE-RAS International Conference on Humanoid Robotics (To appear)*, Santa Monica, Los Angeles, CA, USA., 2004. IEEE Press.

[8] Vassura G. Melchiorri C. Biagiotti L, Tiezzi P. *Modelling and Controlling the Compliance of a Robotic Hand with Soft Finger-pads*. Springer, 2005.

[9] A. Bicchi and G. Tonietti. Fast and soft arm tactics: Dealing with the safety-performance trade-off in robot arms design and control. *IEEE Robotics and Automation Magazine*, 11(2):22–33, 2004.

[10] A. Bicchi, G. Tonietti, M. Bavaro, and M. Piccigallo. Variable stiffness actuators for fast and safe motion control. In B. Siciliano, O. Khatib, and F.C.A. Groen, editors, *Proceedings of ISRR 2003*, Springer Tracts in Advanced Robotics (STAR). Springer Verlag, 2003.

[11] R. Bischoff and V. Graefe. Design principles for dependable robotics assistants. *International Journal of Humanoid Robotics*, 1(1):95–125, 2005.

[12] W. Bluethmann, R. Ambrose, A. Fagg, M. Rosenstein, R. Platt, R. Grupen, C. Brezeal, A. Brooks, A. Lockerd, R. Peters, O. Jenkins, M. Mataric, and M. Bugajska. Building an Autonomous Humanoid Tool User. In *Proceedings of the 2004 IEEE International Conference on Humanoid Robots*, Santa Monica, Los Angeles, CA, USA., 2004. IEEE Press.

[13] Luca Bogoni. Functional features for chopping extracted from observations and interactions. *Image and Vision Computing*, 16:765–783, 1998.

[14] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology.* The MIT Press, Cambridge, Ma, 1984.

[15] C. Breazeal. Social interactions in hri: The robot view. *IEEE Transactions on Man, Cybernetics and Systems: Part C*, 34(2):181–186, 2004.

[16] C. Breazeal, A. Brooks, D. Chilongo, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, and A. Lockerd. Working collaboratively with humanoid robots. In *Proceedings of IEEE-RAS/RSJ International Conference on Humanoid Robots*, Santa Monica, CA, 2004.

[17] C. Breazeal, A. Edsinger, P. Fitzpatrick, and B. Scassellati. Active vision for sociable robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(5):443–453, September 2001.

[18] Cynthia Breazeal. *Sociable Machines: Expressive Social Exchange Between Humans and Robots.* PhD thesis, MIT, Cambridge, Ma, June 2000.

[19] R. Brooks. Challenges for complete creature architectures. In *Proceedings of Simulation of Adaptive Behavior (SAB90)*, 1990.

[20] R. A. Brooks and C. Rosenberg. L - a common lisp for embedded systems. In *Lisp Users and Vendors Conference*, 1995.

[21] Rodney Brooks, Lijin Aryananda, Aaron Edsinger, Paul Fitzpatrick, Charles Kemp, Una-May O'Reilly, Eduardo Torres-Jara, Paulina Varshavskaya, and Jeff Weber. Sensing and manipulating built-for-human environments. *International Journal of Humanoid Robotics*, 1(1), 2004.

[22] Rodney A. Brooks. A robust layered control system for a mobile robot. RA-2:14–23, April 1986.

[23] Rodney A. Brooks. Intelligence without reason. In *Proceedings of the 1991 International Joint Conference on Artificial Intelligence*, pages 569–595, August 1991.

[24] Rodney A. Brooks. *Cambrian Intelligence*. MIT Press, Cambridge, MA, 1999.

[25] J. Burke, R. Murphy, E. Rogers, V. Lumelsky, and J. Scholtz. Final report for the darpa/nsf interdisciplinary study on human-robot interaction. *IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Human-Robot Interaction*, 34(2):103–112, 2004.

[26] Samuel R. Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10(3):37–49, 2005.

[27] S. Calinon and A. Billard. Teaching a humanoid robot to recognize and reproduce social cues. In *Proceedings of the 19th International Symposium on Robot and Human Interactive Communication (RO-MAN06)*, 2006.

[28] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[29] J. G. Cham, S. A Bailey, J. E Clark, R. J Full, and M. R Cutkosky. Fast and robust: Hexapedal robots via shape deposition manufacturing. *The International Journal of Robotics Research*, Volume 21 Issue 10:869–883, 2002.

[30] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, 2001.

[31] Gordon Cheng, Akihiko Nagakubo, and Yasuo Kuniyoshi. Continuous humanoid interaction: An integrated perspective- gaining adaptivity, redundancy, flexibility- in one. *Robotics and Autonomous Systems*, 37(2-3):161–183, November 2001.

[32] Paul Cisek. Neural representations of motor plans, desired trajectories, and controlled objects. *Cognitive Processing*, 6:15–24, 2005.

[33] Johnathan Connell. A behavior-based arm controller. *IEEE Transactions on Robotics and Automation*, 5(5):784–791, December 1989.

[34] Intel Corporation. Open source computer vision library: Reference manual, 2001.

[35] J. Craig. *Introduction to Robotics*. Addison Wesley, second edition edition, 1989.

[36] Aaron M. Dollar and Robert D. Howe. Towards grasping in unstructured environments: Grasper compliance and configuration optimization. *Advanced Robotics*, 19(5):523–543, 2005.

[37] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[38] Zoran Duric, Jeffrey A. Fayman, and Ehud Rivlin. Function from motion. *PAMI*, 18(6):579–591, June 1996.

[39] Exact Dynamics. *The Manus Assistive Robot Manipulator (ARM), http://www.exactdynamics.nl*, 2006.

[40] Yoichiro Endo and Ronald C. Arki. Anticipatory robot navigation by simultaneously localizing and building a cognitive map. In *IROS*, Las Vegas, NV, October 2003.

[41] Andrew W. Fitzgibbon, Geoff Cross, and Andrew Zisserman. Automatic 3d model construction for turn-table sequences. In R. Koch and L. VanGool, editors, *Proceedings of SMILE Workshop on Structure from Multiple Images in Large Scale Environments*, volume 1506 of *Lecture Notes in Computer Science*, pages 154–170. Springer Verlag, June 1998.

[42] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini. Learning About Objects Through Action: Initial Steps Towards Artificial Cognition. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, May 2003.

[43] Paul Fitzpatrick. *From First Contact to Close Encounters: A developmentally deep perceptual system for a humanoid robot*. PhD thesis, Massachusetts Institute of Technology, 2003.

[44] Paul Fitzpatrick, Giorgio Metta, Lorenzo Natale, Sajit Rao, and Giulio Sandini. Learning about objects through action - initial steps towards artificial cognition. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA-03)*, volume 3. IEEE Press, 2003.

[45] Paul Fitzpatrick and Eduardo Torres-Jara. The power of the dark side: using cast shadows for visually-guided reaching. In *Proceedings of the International Conference on Humanoid Robotics*, 2004.

[46] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42:143–166, 2002.

[47] D. A. Forsyth and Jean Ponce. *Computer Vision: a modern approach*. Prentice Hall, 2002.

[48] Chris Gaskett and Gordon Cheng. Online learning of a motor map for humanoid robot reaching. In *Proceedings of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Singapore, December 2003.

[49] M. Gentilucci, A. Roy, and S. Stefanini. Grasping an object naturally or with a tool: are these tasks guided by a common motor representation? *Experimental Brain Research*, 157(4):496–506, Auguest 2004.

[50] J. J. Gibson. *The ecological approach to visual perception AUTHORS: Gibson, J. J. (1979)*. Houghton Mifflin, 1979.

[51] M. R. Goulding, M. E. Rogers, and S. M. Smith. Public health and aging: Trends in aging - united states and worldwide. *The Journal of the American Medical Association*, 289:1371–1373, 2003.

[52] Richard Gourdeau. *ROBOOP - A robotics object oriented package in C+.* http://www.cours.polymtl.ca/roboop/, 2005.

[53] Jeffrey A. Guertin and William T. Townsend. Teleoperator slave - wam design methodology. *Industrial Robot*, 26:167–177, 1999.

[54] Karen Haigh and Holly A. Yanco. Automation as caregiver: A survey of issues and technologies. In *Proceedings of the AAAI-2002 Workshop on Automation as Caregiver: The Role of Intelligent Technology in Elder Care*, Edmonton, Alberta, 2002.

[55] S. Hart, S. Ou, J. Sweeney, , and R. Grupen. A framework for learning declarative structure. In *Proceedings of the Robotics, Science & Systems Workshop on Manipulation for Human Environments*, Philedelphia, Pennsylvania, August 2006.

[56] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[57] Kuzanuao Uchiyama Haruhisa Kawasaki, Tsuneo Komatsu. Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu hand ii. *IEEE Transactions on Mechatronics*, 7(3):296–303, September 2002.

[58] M. Hillman, K. Hagan, S. Hagan, J. Jepson, and R. Orpwood. A wheelchair mounted assistive robot. In *Proceedings of ICORR '99: International Conference on Rehabilitation Robotics*, 1999.

[59] G. Hirzinger, N. Sporer, A. Albu-Schaffer, M. Hahnle, and A. Pascucci. Dlr?s torque-controlled light weight robot iii?are we reaching the technological limits now? In *Proceedings of the International Conference on Robotics Automation*, page 1710?1716, 2002.

[60] Ian Horswill. Polly: A vision-based artificial agent. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 824–829, Menlo Park, CA, USA, 1993. AAAI Press.

[61] E. Huber and K. Baker. Using a hybrid of silhouette and range templates for real-time pose estimation. In *Proceedings of ICRA 2004 IEEE International Conference on Robotics and Automation*, volume 2, pages 1652–1657, 2004.

[62] R. A. Peters II, K. E. Hambuchen, K. Kawamura, and D. M. Wilkes. The sensory ego-sphere as a shortterm memory for humanoids. In *IEEE-RAS International Conference on Humanoid Robots*, pages 451–459, Tokyo, Japan, November 2001.

[63] F. Iida and R. Pfeifer. Sensing through body dynamics. *Robotics and Autonomous Systems*, 54(8):631–640, August 2006.

[64] InterSense Inc. *Intersense InertiaCube3 Manual*. "http://www.isense.com", 2006.

[65] Point Grey Research Inc. *FireFly2 IEEE-1394 CCD Camera Manual*. "http://www.ptgrey.com/products/firefly2/firefly2.pdf", 2004.

[66] H. Inoue. Force feedback in precise assembly tasks. In P.H. Winston and R.H. Brown, editors, *Artificial Intelligence: An MIT Perspective*. The MIT Press, 1979.

[67] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(11):1254–1259, 1998.

[68] P. Dilworth J. Pratt, M. Chew and G. Pratt. Virtual Model Control: An Intuitive Approach for Bipedal Locomotion. *International Journal of Robotics Research*, 20(2):129–143, 2001.

[69] M. Jagersand and R. Nelson. Visual Space Task Specification, Planning and Control. In *Proceedings of the IEEE International Symposium on Computer Vision*, pages 521–526, 1995.

[70] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[71] E. R. Kandel, J. H. Schwartz, and T. Jessell. *Principles of Neural Science*. McGraw-Hill and Appleton and Lange, New York, NY, 4th edition, January 2000.

[72] Charles C. Kemp. *A Wearable System that Learns a Kinematic Model and Finds Structure in Everyday Manipulation by using Absolute Orientation Sensors and a Camera*. PhD thesis, Massachusetts Institute of Technology, May 2005.

[73] Charles C. Kemp. *pysense: A python based robotics package*. http://robotmanipulation.org/pysense/, 2006.

[74] Charles C. Kemp and Aaron Edsinger. Visual Tool Tip Detection and Position Estimation for Robotic Manipulation of Unknown Human Tools. Technical Report AIM-2005-037, MIT Computer Science and Artificial Intelligence Laboratory, 2005.

[75] Charles C. Kemp and Aaron Edsinger. Robot manipulation of human tools: Autonomous detection and control of task relevant features. In *In Submission to: 5th IEEE International Conference on Development and Learning (ICDL-06)*, Bloomington, Indiana, 2006.

[76] Charles C. Kemp and Aaron Edsinger. What can i control?: The development of visual categories for a robot's body and the world that it influences. In *In Submission to Fifth International Conference on Development and Learning, Special Session on Autonomous Mental Development*, 2006.

[77] O. Khatib. A unified approach to motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics and Automation*, 3(1):43–53, 1987.

[78] O. Khatib, Brock O. Chang K. Yokoi, K., and A. Casal. Robots in human environments: Basic autonomous capabilities. *International Journal of Robotics Research*, 18(684), 1999.

[79] H. Kikuchi, M. Yokoyama, K. Hoashi, Y. Hidaki, T. Kobayashi, and K. Shirai. Controlling gaze of humanoid in communication with human. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, B.C., Canada, October 1998.

[80] H. Kozima and H. Yano. A robot that learns to communicate with human caregivers. In *Proceedings of the International Workshop on Epigentic Robotics*, 2001.

[81] D. Kragic and H. I. Chrisensen. Survey on visual servoing for manipulation. Technical report, Computational Vision and Active Perception Laboratory, 2002.

[82] K. Kyong, S. Freedman, M. Mataric, M. Cunningham, and B. Lopez. A hands-off physical therapy assistance robot for cardiac patients. In *9th International Conference on Rehabilitation Robotics*, pages 337– 340, July 2005.

[83] Dale A. Lawrence. Actuator limitations on achievable manipulator impedance. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, 1989.

[84] E. Littman, A. Drees, and H. Ritter. Robot guidance by human pointing gestures. In *Proceedings of th International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing*, 1996.

[85] C. Lovchik and M. Diftler. The robonaut hand: A dexterous robot hand for space. In *Proceedings of the IEEE International Conference on Automation and Robotics*, volume 2, pages 907–912, Detroit, Michigan, May 1999.

[86] D.N. Ly, K. Regenstein, T. Asfour, and R. Dillmann. A modular and distributed embedded control architecture for humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Japan, September 2004.

[87] Mon-Williams M.1 and Tresilian J.R. A simple rule of thumb for elegant prehension. *Current Biology*, 11(13):1058–1061, July 2001.

[88] Ikuo Yamano Takashi Maeno. Five-fingered robot hand using ultrasonic motors elastic elements. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[89] Matthew Mason. *Mechanics of Robotic Manipulation*. MIT Press, August 2001. Intelligent Robotics and Autonomous Agents Series, ISBN 0-262-13396-2.

[90] M. Mataric. Behavior-based control: Main properties and implications. In *Proceedings of Workshop on Intelligent Control Systems, International Conference on Robotics and Automation*, Nice, France, May 1992.

[91] G. Metta, P. Fitzpatrick, and L. Natale. Yarp: yet another robot platform. *International Journal on Advanced Robotics Systems. Special Issue on Software Development and Integration in Robotics*, 3(1):43–48, March 2006.

[92] Giorgio Metta. *Babybot: a study into sensorimotor development.* PhD thesis, LIRA-Lab, DIST, University of Genoa, 2000.

[93] Giorgio Metta and Paul Fitzpatrick. Early integration of vision and manipulation. *Adaptive Behavior*, 11(2):109–128, June 2003.

[94] Michel, Gold, and Scassellati. Motion-based robotic self-recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[95] Y. Nagai. Learning to comprehend deictic gestures in robots and human infants. In *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication, ROMAN 2005*, pages 217–212, August 2005.

[96] L. Natale, G. Metta, and G. Sandini. Learning haptic representations of objects. In *In International Conference on Intelligent Manipulation and Grasping*, Genoa, Italy, July 2004.

[97] Lorenzo Natale. *Linking Action to Perception in a Humanoid Robot: A Developmental Approach to Grasping.* PhD thesis, LIRA-Lab, DIST, University of Genoa, 2004.

[98] D. A. Norman. *The Design of Everyday Things.* Doubleday, New York.

[99] Peng Pan, Kevin Lynch, and Michael A. Peshkin. Human interaction with passive assistive robots. In *IEEE 9th International Conference on Rehabilitation Robotics*, June 2005.

[100] F. Panerai, G. Metta, and G.Sandini. Learning vor-like stabilization reflexes in robots. In *8th European Symposium on Artificial Neural Networks (ESANN 2000)*, Bruges, Belgium, April 2000.

[101] R. Pfeifer. and G. Gómez. Interacting with the real world: design principles for intelligent systems. *Artificial life and Robotics*, 9.:1–6, 2005.

[102] Justus H. Piater and Roderic A. Grupen. Learning appearance features to support robotic manipulation. *Cognitive Vision Workshop*, 2002.

[103] R. Platt, A. Fagg, and R. Grupen. Nullspace composition of control laws for grasping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[104] R. Platt, A. H. Fagg, and R. Grupen. Manipulation gaits: Sequences of grasp control tasks. In *Proceedings of the 2004 IEEE Conference on Robotics and Automation (ICRA)*, April 2004.

[105] Robert Platt, Oliver Brock, Andrew H. Fagg, Deepak R. Karuppiah, Michael T. Rosenstein, Jefferson A. Coelho Jr., Manfred Huber, Justus H. Piater, David Wheeler, and Roderic A. Grupen. A Framework For Humanoid Control and Intelligence. In *Proceedings of the 2003 IEEE International Conference on Humanoid Robots*, 2003.

[106] M. Pollack, S. Engberg, J.T. Matthews, Sebastian Thrun, L. Brown, D. Colbry, C. Orosz, B. Peintner, S. Ramakrishnan, J. Dunbar-Jacob, C. McCarthy, Michael Montemerlo, Joelle Pineau, and Nicholas Roy. Pearl: A mobile robotic assistant for the elderly. In *Workshop on Automation as Caregiver: the Role of Intelligent Technology in Elder Care (AAAI)*, August 2002.

[107] N. Pollard and J.K. Hodgins. Generalizing Demonstrated Manipulation Tasks. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics (WAFR '02)*, December 2002.

[108] G. Pratt and M. Williamson. Series Elastic Actuators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-95)*, volume 1, pages 399–406, Pittsburg, PA, July 1995.

[109] T. Asfour R. Zöllner and R. Dillmann. Programming by demonstration: Dual-arm manipulation tasks for humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, Sendai, Japan, September 28 - October 2, 2004 2004.

[110] R.G. Radwin and J.T. Haney. An ergonomics guide to hand tools. Technical report, American Institutional Hygiene Association, 1996. http://ergo.engr.wisc.edu/pubs.htm.

[111] Marc H. Raibert. *Legged robots that balance*. Massachusetts Institute of Technology, Cambridge, MA, USA, 1986.

[112] David Robinson. *Design and Analysis of Series Elasticity in Closed-loop Actuator Force Control*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2000.

[113] Erich Rome, Joachim Hertzberg, Georg Dorffner, and Patrick Doherty. 06231 executive summary – towards affordance-based robot control. In Erich Rome, Patrick Doherty, Georg Dorffner, and Joachim Hertzberg, editors, *Towards Affordance-Based Robot Control*, number 06231 in Dagstuhl Seminar Proceedings, 2006.

[114] M. Rosenstein, A. Fagg, S. Ou, and R. Grupen. User intentions funneled through a human-robot interface. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pages 257–259, 2005.

[115] J. Rosier, J. van Woerden, L. van der Kolk, B. Driessen, H. Kwee, J. Duimel, J. Smits, A. Tuinhof de Moed, G. Honderd, and P. Bruyn. Rehabilitation robotics: The manus concept. In *Proceedings of the Fifth International Conference on Advanced Robotics: Robots in Unstructured Environments*, volume 1, 1991.

[116] M. Rucci and G. Desbordes. Contributions of fixational eye movements to the discrimination of briefly presented stimuli. *Journal of Vision*, 3(11):852–864, 2003.

[117] M. Saha and P. Isto. Motion planning for robotic manipulation of deformable linear objects. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006.

[118] F. Santini and M. Rucci. Active estimation of distance in a robotic system that replicates human eye movements. *Journal of Robotics and Autonomous Systems*, In Press, 2006.

[119] N. Sian, K. Yoki, Y. Kawai, and K. Muruyama. Operating humanoid robots in human environments. In *Proceedings of the Robotics, Science & Systems Workshop on Manipulation for Human Environments*, Philedelphia, Pennsylvania, August 2006.

[120] T. Simeon, J. P. Laumond, J. Cortes, and A. Sahbani. Manipulation planning with probabilistic roadmaps. *IJRR*, 23(7-8):729–746, 2004.

[121] Beate Sodian and Claudia Thoermer. Infants' understanding of looking, pointing, and reaching as cues to goal-directed action. *Journal of Cognition and Development*, 5(3):289–316, 2004.

[122] W. Song, H. Lee, J. Kim, Y. Yoon, and Z Bien. Kares: Intelligent robotics system for the disabled and the elderly. In *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Socieyt*, volume 20, 1998.

[123] R St. Amant and A.b Wood. Tool use for autonomous agents. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 184–189, 2005.

[124] R. Stiefelhagen, C. Fugen, P. Gieselmann, H. Holzapfel, K. Nickel, and A. Waibel. Natural human-robot interaction using speech, head pose and gestures. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, September 2004.

[125] C. Stiller and J. Konrad. Estimating motion in image sequences, a tutorial on modeling and computation of 2d motion. *IEEE Signal Process. Mag., vol. 16, pp. 70–91*, 1999.

[126] Alexandar Stoytchev. Behavior-grounded representation of tool affordances. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[127] Adriana Tapus and Maja J. Mataric´. Towards socially assistive robotics. *International Journal of the Robotics Society of Japan*, 24(5), 2006.

[128] Geoffrey Taylor. *Robust Perception and Control for Humanoid Robots in Unstructured Environments Using Vision.* PhD thesis, Monash University, Clayton, Victoria 3800, Australia, February 2004.

[129] Russell L Tedrake. *Applied Optimal Control for Dynamically Stable Legged Locomotion.* PhD thesis, Massachusetts Institute of Technology, 2004.

[130] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics.* MIT Press, 2005.

[131] Emanuel Todorov and Michael Jordan. Optimal feedback control as as theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235, 2002.

[132] Deepak Tolani, Ambarish Goswami, and Norman I. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, 62(5):353–388, 2000.

[133] A. Torralba. Modeling global scene factors in attention. *Journal of Optical Society of America A: Special Issue on Bayesian and Statistical Approaches to Vision*, 20(7):1407–1418, 2003.

[134] Eduardo Torres-Jara. Obrero: a platform for sensitive manipulation dec. 5, 2005 page(s):327 - 332. In *5th IEEE-RAS International Conference on Humanoid Robots*, pages 327 – 332, 2005.

[135] W. T. Townsend and Salisbury. *"Robots and biological systems : towards a new bionics?"*, chapter Mechanical design for wholearm manipulation. Springer-Verlag, 1993.

[136] Karl T. Ulrich, Timothy D. Tuttle, Joseph P. Donoghue, and William T. Townsend. Intrinsically safer robots. Technical report, Barrett Technologies and NASA Kennedy Space Center, Cambridge, MA. USA, 1995.

[137] J. Versace. A review of the severity index. In *Proceedings of the 15th Stapp Car Crash Conference*, pages 771–796, 1971.

[138] P. Viola and M. Jones. Robust real-time object detection. 57(2):137–154, 2004.

[139] B. Waarsing, M. Nuttin, and H. Van Brussel. Introducing robots into a human-centred environment - the behaviour-based approach. In *Proceedings of the 4th International Conference on CLAWAR*, pages 465–470, May 2001.

[140] Z. Wasik and A. Saffiotti. A hierarchical behavior-based approach to manipulation tasks. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA-03)*. IEEE Press, 2003.

[141] D.E. Whitney and J.L. Nevins. What is the remote centre compliance (rcc) and what can it do? In *Proceedings of the 9th International Symposium on Robots*, pages 135–152, Washington, D.C., 1979.

[142] D. Williams and O. Khatib. The virtual linkage: A model for internal forces in multi-grasp manipulation. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, page 1025?30, 1993.

[143] M. Williamson. *Robot Arm Control Exploiting Natural Dynamics*. PhD thesis, Massachusetts Institute of Technology, 1999.

[144] Matthew M. Williamson. Exploiting natural dynamics in robot control. In *Fourteenth European Meeting on Cybernetics and Systems Research (EMCSR '98)*, Vienna, Austria, 1998.

[145] Matthew M. Williamson. Rhythmic robot control using oscillators. In *IROS '98*, 1998. Submitted.

[146] Patrick H. Winston, Boris Katz, Thomas O. Binford, and Michael R. Lowry. Learning physical descriptions from functional definitions, examples, and precedents. In *National Conference on Artificial Intelligence*, pages 433–439, 1983.

[147] J.M. Wolfe. Guided search 2.0: a revised model of visual search. *Psychonomic Bulletin and Review*, 1(3):202–238, 1994.

[148] Tom Worsnopp, Michael A. Peshkin, Kevin Lynch, and J. Edward Colgate. Controlling the apparent inertia of passive human-interactive robots. *Journal of Dynamic Systems Measurement and Control*, 128(1), March 2006.

[149] S. Yigit, C. Burghart, and H. Worn. Specific combined control mechanisms for human robot co-operation. In *Proceedings of ISSC*, San Diego, California, 2003.

[150] S. Yigit, C. Burghart, and H. Wörn. Concept of combined control mechanisms for human-robot-co-operation. In *Proceedings of the International Conference on Computer, Communication and Control Technologies (CCCT)*, Orlando, Florida, 2003.

[151] K. Yokohama, H. Handa, T. Isozumi, Y. Fukase, K. Kaneko, F Kanehiro, Y. Kawai, F. Tomita, and H. Hirukawa. Cooperative works by a human and a humanoid robot. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, September 2003.

[152] M. Zinn, O. Khatib, B. Roth, and J. K. Salisbury. A new actuation approach for human friendly robot design. In *International Symposium on Experimental Robotics*, Italy, July 2002.

[153] M. Zinn, O. Khatib, B. Roth, and J.K. Salisbury. Playing it safe: human-friendly robots. *IEEE Robotics & Automation Magazine*, 11(2):12–21, June 2004.